Decomposition in conic optimization with partially separable structure

Sun, Yifan; Andersen, Martin Skovgaard; Vandenberghe, Lieven

Published in:
S I A M Journal on Optimization

Link to article, DOI:
10.1137/130926924

Publication date:
2014

Document Version
Peer reviewed version

DECOMPOSITION IN CONIC OPTIMIZATION WITH PARTIALLY SEPARABLE STRUCTURE

YIFAN SUN*, MARTIN S. ANDERSEN†, AND LIEVEN VANDENBERGHE*

Abstract. Decomposition techniques for linear programming are difficult to extend to conic optimization problems with general non-polyhedral convex cones because the conic inequalities introduce an additional nonlinear coupling between the variables. However in many applications the convex cones have a partially separable structure that allows them to be characterized in terms of simpler lower-dimensional cones. The most important example is sparse semidefinite programming with a chordal sparsity pattern. Here partial separability derives from the clique decomposition theorems that characterize positive semidefinite and positive-semidefinite-completable matrices with chordal sparsity patterns. The paper describes a decomposition method that exploits partial separability in conic linear optimization. The method is based on Spingarn's method for equality constrained convex optimization, combined with a fast interior-point method for evaluating proximal operators.

Key words. semidefinite programming, decomposition, interior-point algorithms

AMS subject classifications. 90C22, 90C25, 90C51

1. Introduction.

We consider conic linear optimization problems (conic LPs)

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \mathcal{C} \end{array} \tag{1.1}$$

in which the cone $\mathcal{C}$ is defined in terms of lower-dimensional convex cones $\mathcal{C}_k$ as

$$\mathcal{C} = \{x \in \mathbf{R}^n \mid x_{\gamma_k} \in \mathcal{C}_k,\ k = 1, \ldots, l\}. \tag{1.2}$$

The sets γ_k are ordered subsets of $\{1, 2, \ldots, n\}$ and x_{γ_k} denotes the subvector of x with entries indexed by γ_k. We refer to the structure in the cone $\mathcal{C}$ as *partial separability*. The purpose of the paper is to describe a decomposition method that exploits partially separable structure.

In standard linear optimization, with $\mathcal{C} = \mathbf{R}^n_+$, the cone is separable, *i.e.*, a product of one-dimensional cones, and the coupling of variables and constraints is entirely specified by the sparsity pattern of A. The term *decomposition* in linear optimization usually refers to techniques for exploiting angular or dual-angular structure in the coefficient matrix A, *i.e.*, a sparsity pattern that is almost block-diagonal, except for a small number of dense rows or columns [28]. The goal of a decomposition algorithm is to solve the problem iteratively, by solving a sequence of separable problems, obtained by removing the complicating variables or constraints. The decoupled subproblems can be solved in parallel or sequentially (for example, to reduce memory usage). Moreover, if the iterative coordinating process is simple enough to be decentralized, the decomposition method can be used as a distributed algorithm. By extension, decomposition methods can be applied to more general sparsity patterns for which removal of complicating variables and constraints makes the problem substantially easier to solve (even if it does not decompose into independent subproblems).

*Electrical Engineering Department, UCLA. Email: ysun01@ucla.edu, vandenbe@ucla.edu. Research supported by NSF Grants DMS-1115963 and ECCS-1128817.

†Department of Applied Mathematics and Computer Science, Technical University of Denmark, Email: mskan@dtu.dk.

When the cone $\mathcal{C}$ in (1.1) is not separable or block-separable (a product of lower-dimensional cones), the formulation of decomposition algorithms is more complicated because the inequalities introduce an additional coupling between the variables. However if the cone is partially separable, as defined in (1.2), and the overlap between the index sets γ_k is small, one can formulate efficient decomposition algorithms by introducing auxiliary independent optimization variables for the subvectors x_{γ_k} and adding equality constraints to impose consistency between the variables that refer to the same components of x. The decomposition method discussed in this paper follows this conversion approach and solves the reformulated problem by a first-order splitting method. The quadratic conic subproblems that need to be solved at each iteration are solved by a customized interior-point method. The method is described in sections 2 and 3.

An important example of a partially separable cone is the cone of positive-semidefinite-completable sparse matrices with a chordal sparsity pattern. Matrices in this cone are characterized by the property that all their principal dense submatrices are positive semidefinite [21, theorem 7]. This fundamental result has been applied in previous methods for sparse semidefinite optimization. It is the basis of the conversion methods used to reformulate sparse semidefinite programs (SDPs) in equivalent forms that are easier to handle by interior-point algorithms [25, 16] or more suitable for distributed algorithms via the Alternating Direction Method of Multipliers (ADMM) [13]. Applied to sparse semidefinite programs the conversion method mentioned in the previous paragraph reduces to the clique-tree conversion methods developed in [25, 16]. Partial separability also underlies the saddle-point mirror-prox algorithm for 'well-structured' sparse SDPs proposed in [30]. We discuss the sparse semidefinite optimization application of the decomposition method in detail in sections 4 and 5, and present numerical results in section 6.

Notation. If α is a subset of $\{1, 2, \ldots, n\}$, then E_α denotes the $|\alpha| \times n$-matrix with entries $(E_\alpha)_{ij} = 1$ if $\alpha(i) = j$ and $(E_\alpha)_{ij} = 0$ otherwise. Here $\alpha(i)$ is the ith element of α, sorted using the natural ordering. If not explicitly stated the column dimension n of E_α will be clear from the context. The result of multiplying an n-vector x with E_α is the subvector of x of length $|\alpha|$ with elements $(x_\alpha)_k = x_{\alpha(k)}$. The adjoint operation $x = E_\alpha^T y$ maps an $|\alpha|$-vector y to an n-vector x by copying the entries of y to the positions indicated by α, *i.e.*, by setting $x_{\alpha(k)} = y_k$ and $x_i = 0$ for $i \notin \alpha$. Therefore $E_\alpha E_\alpha^T$ is an identity matrix of order $|\alpha|$ and $E_\alpha^T E_\alpha$ is a diagonal 0-1 matrix of order n, with ith diagonal entry equal to one if and only if $i \in \alpha$. The matrix $P_\alpha = E_\alpha^T E_\alpha$ represents projection in $\mathbf{R}^n$ on the sparse n-vectors with support α. Similar notation will be used for principal submatrices in a symmetric matrix. If $X \in \mathbf{S}^p$ (the symmetric matrices of order p) and α is a subset of $\{1, \ldots, p\}$, then $\mathcal{E}_\alpha(X) = X_{\alpha\alpha} = E_\alpha X E_\alpha^T \in \mathbf{S}^{|\alpha|}$. This is the submatrix of order $|\alpha|$ with i, j entry $(X_{\alpha\alpha})_{ij} = X_{\alpha(i)\alpha(j)}$. The adjoint $\mathcal{E}_\alpha^*$ copies a matrix $Y \in \mathbf{S}^{|\alpha|}$ to an otherwise zero symmetric $p \times p$-matrix: $\mathcal{E}_\alpha^*(Y) = E_\alpha^T Y E_\alpha$. The projection of a matrix $X \in \mathbf{S}^p$ on the matrices that are zero outside of a diagonal $\alpha \times \alpha$ block is denoted $\mathcal{P}_\alpha(X) = P_\alpha X P_\alpha$.

2. Partially separable cones.

2.1. Partial separability. A function $f : \mathbf{R}^n \to \mathbf{R}$ is *partially separable* if it can be expressed as $f(x) = \sum_{k=1}^{l} f_k(A_k x)$, where each A_k has a nontrivial nullspace, *i.e.*, a rank substantially less than n. This concept was introduced by Griewank and Toint [19, 20]. Here we consider the simplest and most common example of partial separability and assume that $A_k = E_{\gamma_k}$ for some index set $\gamma_k \subset \{1, 2, \ldots, n\}$. This

means that f can be written as a sum of functions that depend only on subsets of the components of x: $f(x) = \sum_{k=1}^{l} f_k(x_{\gamma_k})$. Partial separability generalizes *separability* ($l = n$, $\gamma_k = \{k\}$) and *block-separability* (the sets γ_k form a partition of $\{1, 2, \ldots, n\}$).

We call a cone $\mathcal{C} \subset \mathbf{R}^n$ *partially separable* if it can be expressed as

$$\mathcal{C} = \{x \mid E_{\gamma_k} x \in \mathcal{C}_k,\ k = 1, \ldots, l\} \tag{2.1}$$

where $\mathcal{C}_k$ is a convex cone in $\mathbf{R}^{|\gamma_k|}$. The terminology is motivated by the fact the indicator function of $\mathcal{C}$ is a partially separable function: $\delta_{\mathcal{C}}(x) = \sum_{k=1}^{l} \delta_{\mathcal{C}_k}(E_{\gamma_k} x)$ where $\delta_{\mathcal{S}}$ is the indicator function of a set $\mathcal{S}$. The following assumptions are made.

- The index sets γ_k are *distinct* and *maximal*, *i.e.*, $\gamma_i \not\subseteq \gamma_j$ for $i \neq j$, and their union is equal to $\{1, 2, \ldots, n\}$.
- The convex cones $\mathcal{C}_k$ are proper, *i.e.*, closed, pointed, with nonempty interior. This implies that their dual cones $\mathcal{C}_k^* = \{v \in \mathbf{R}^{|\gamma_k|} \mid u^T v \geq 0\ \forall u \in \mathcal{C}_k\}$ are proper convex cones and that $\mathcal{C}_k = \mathcal{C}_k^{**}$.
- There exists a point $\bar{x}$ with $E_{\gamma_k} \bar{x} \in \mathbf{int}\, \mathcal{C}_k$ for $k = 1, \ldots, l$.

These assumptions imply that $\mathcal{C}$ is itself a proper cone. It is closed because it can be expressed as an intersection of closed halfspaces: $x \in \mathcal{C}$ if and only if $u_k^T E_{\gamma_k} x \geq 0$ for all $u_k \in \mathcal{C}_k^*$, $k = 1, \ldots, l$. The cone $\mathcal{C}$ is pointed because $x \in \mathcal{C}$, $-x \in \mathcal{C}$ implies $E_{\gamma_k} x \in \mathcal{C}_k$ and $-E_{\gamma_k} x \in \mathcal{C}_k$ for all k. Since the cones $\mathcal{C}_k$ are pointed, we have $E_{\gamma_k} x = 0$ for $k = 1, \ldots, l$. Since the index sets γ_k cover $\{1, 2, \ldots, n\}$, this implies $x = 0$. The cone $\mathcal{C}$ has nonempty interior because the point $\bar{x}$ is in its interior.

The dual cone of $\mathcal{C}$ is

$$\mathcal{C}^* = \{\sum_{k=1}^{l} E_{\gamma_k}^T \tilde{s}_k \mid \tilde{s}_k \in \mathcal{C}_k^*,\ k = 1, \ldots, l\}. \tag{2.2}$$

To show this, we denote the set on the right-hand side of (2.2) by $\mathcal{K}$ and first note that $\mathcal{C} = \mathcal{K}^*$, *i.e.*, $\mathcal{C}$ is the dual cone of $\mathcal{K}$. Indeed, a vector x satisfies $x^T s \geq 0$ for all $s \in \mathcal{K}$ if and only if $x^T E_{\gamma_k}^T \tilde{s}_k \geq 0$ for all $\tilde{s}_k \in \mathcal{C}_k^*$ and all k. This condition is equivalent to $E_{\gamma_k} x \in \mathcal{C}_k$ for $k = 1, \ldots, l$, *i.e.*, $x \in \mathcal{C}$. If $\mathcal{C}$ is the dual cone of $\mathcal{K}$, then $\mathcal{K}$ is the closure of the dual cone of $\mathcal{C}$, and to establish (2.2) it remains to show that $\mathcal{K}$ is closed. This follows from the third assumption which implies the property

$$\sum_k E_{\gamma_k}^T \tilde{s}_k = 0,\ \ \tilde{s}_k \in \mathcal{C}_k^*,\ \ k = 1, \ldots, l \qquad \Longrightarrow \qquad \tilde{s}_k = 0,\ \ k = 1, \ldots, l. \tag{2.3}$$

This can be seen by taking the inner product of $\bar{x}$ and the sum on the left-hand side: since $E_{\gamma_k} \bar{x} \in \mathbf{int}\, C_k$ we have $\sum_{k=1}^{l} \bar{x}^T E_{\gamma_k}^T \tilde{s}_k = 0$ for $\tilde{s}_k \in \mathcal{C}_k^*$ only if $\tilde{s}_k = 0$ for all k. The property (2.3) is a sufficient condition for closedness of $\mathcal{K}$ (see theorem 9.1 or its corollary 9.1.3 in [36]). We conclude that $\mathcal{K}$ is closed and $\mathcal{K} = \mathcal{C}^*$.

2.2. Sparsity and intersection graph. Two useful undirected graphs can be associated with the partially separable structure defined by the index sets γ_k. These graphs will be referred to as the *sparsity graph* and the *intersection graph*. The sparsity graph $\mathcal{G}$ has n vertices, representing the n variables. There is an edge between two distinct vertices i and j if $i,\ j \in \gamma_k$ for some k. We call this the sparsity graph because it represents the sparsity pattern of a matrix $H = \sum_{k=1}^{l} E_{\gamma_k}^T H_k E_{\gamma_k}$ where the matrices H_k are dense symmetric matrices. The entries $(i, j) \notin \cup_{k=1,\ldots,l} (\gamma_k \times \gamma_k)$ are the positions of the zeros in the sparsity pattern of H. Each index set γ_k thus defines a complete subgraph of the sparsity graph $\mathcal{G}$. Since the index sets are maximal

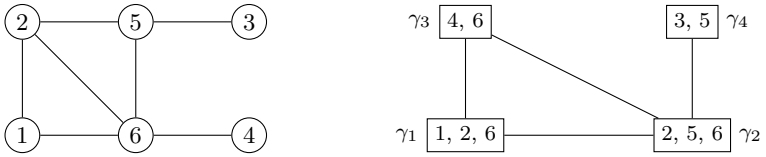

FIG. 2.1. *Sparsity graph and intersection graph for an example with $n = 6$ and four index sets $\gamma_1 = \{1, 2, 6\}$, $\gamma_2 = \{2, 5, 6\}$, $\gamma_3 = \{4, 6\}$, $\gamma_4 = \{3, 5\}$.*

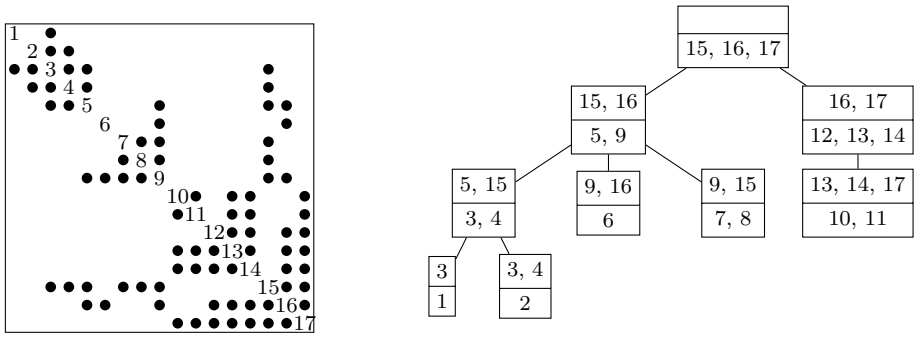

FIG. 2.2. *Spanning tree in an intersection graph for nine index sets γ_k and $n = 17$ variables (right), and the sparsity pattern for the corresponding sparsity graph (left). The sparsity graph is chordal. Each vertex γ_k in the spanning tree is split in two sets α_k and $\gamma_k \setminus \alpha_k$ with α_k the intersection of γ_k and its parent. The indices listed in the top row of each vertex form α_k. The indices in the bottom row form $\gamma_k \setminus \alpha_k$.*

(by assumption), these complete subgraphs are the cliques in $\mathcal{G}$. (In this paper, we use the term clique for a *maximal* complete subgraph.)

The intersection graph has the index sets γ_k as its vertices and an edge between distinct vertices i and j if the sets γ_i and γ_j intersect. We place a weight $|\gamma_i \cap \gamma_j|$ on edge $\{i, j\}$. The intersection graph is therefore identical to the clique graph of the sparsity graph $\mathcal{G}$. (The clique graph of an undirected graph has the cliques of the graph as its vertices and undirected edges between cliques that intersect, with edge weights equal to the sizes of the intersection.) An example is shown in Figure 2.1.

2.3. Chordal structure. An undirected graph is *chordal* if for every cycle of length greater than three there is a chord (an edge connecting non-consecutive vertices in the cycle). If the sparsity graph representing a partially separable structure is chordal (as will be the case in the application to semidefinite optimization discussed in the second half of the paper), several additional useful properties hold.

A spanning tree of the intersection graph (or, more accurately, a spanning forest, since we do not assume the intersection graph is connected) has the *running intersection property* if $\gamma_i \cap \gamma_j \subseteq \gamma_k$ whenever vertex γ_k is on the path between vertices γ_i and γ_j in the tree. A fundamental theorem states that a spanning tree with the running intersection property exists if and only if the corresponding sparsity graph is chordal [8]. The right-hand figure in Figure 2.2 shows a spanning tree of the intersection graph of $l = 9$ index sets γ_k with $n = 17$ variables. On the left-hand side we represent the corresponding sparsity graph as a sparse matrix pattern (a dot in positions i, j and j, i indicates an edge $\{i, j\}$). It can be verified that the tree satisfies the running intersection property.

Now suppose we partition each index set γ_k in two sets α_k and $\gamma_k \setminus \alpha_k$, defined as

follows. If γ_k is the root of the tree (or a root if it is a forest), then $\alpha_k = \emptyset$. For the other vertices, $\alpha_k = \gamma_k \cap \gamma_{\mathrm{pa}(\gamma_k)}$ where $\mathrm{pa}(\gamma_k)$ is the parent of γ_k in the tree. Then the running intersection property has the important implication that the sets $\gamma_k \setminus \alpha_k$ form a partition of $\{1, 2, \ldots, n\}$ [29, 34]. Figure 2.2 illustrates the definition of α_k. For more background on chordal graphs we refer the reader to survey paper [8].

3. Conic optimization with partially separable cones. We now consider a pair of conic linear optimization problems

$$\begin{array}{lll lll} \text{minimize} & c^T x & & \text{maximize} & b^T y \\ \text{subject to} & Ax = b & & \text{subject to} & A^T y + s = c \\ & x \in \mathcal{C} & & & s \in \mathcal{C}^* \end{array} \tag{3.1}$$

with respect to a partially separable cone (2.1) and its dual (2.2). The variables are $x, s \in \mathbf{R}^n$, $y \in \mathbf{R}^m$. In addition to the assumptions listed in section 2.1 we assume that the sparsity graph associated with the index sets γ_k is chordal and that a maximum weight spanning tree (or forest) T in the intersection graph is given. We refer to T as the *intersection tree* and use the notation $\mathrm{pa}(\gamma_k)$ and $\mathrm{ch}(\gamma_k)$ for the parent and the children of vertex γ_k in T.

3.1. Reformulation. The decomposition algorithm developed in the following sections is based on a straightforward reformulation of the conic LPs (3.1). The primal and dual cones can be expressed as $\mathcal{C} = \{x \mid Ex \in \tilde{\mathcal{C}}\}$ and $\mathcal{C}^* = \{E^T \tilde{s} \mid \tilde{s} \in \tilde{\mathcal{C}}^*\}$, where $\tilde{\mathcal{C}} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_l$, $\tilde{\mathcal{C}}^* = \mathcal{C}_1^* \times \cdots \times \mathcal{C}_l^*$, and E is the $\tilde{n} \times n$ matrix

$$E = \begin{bmatrix} E_{\gamma_1}^T & E_{\gamma_2}^T & \cdots & E_{\gamma_l}^T \end{bmatrix}^T \tag{3.2}$$

with $\tilde{n} = \sum_k |\gamma_k|$. Define $\mathcal{V} = \mathrm{Range}(E)$. A change of variables $\tilde{x} = Ex$, $s = E^T \tilde{s}$ allows us to write the problems (3.1) equivalently as

$$\begin{array}{lll lll} \text{minimize} & \tilde{c}^T \tilde{x} & & \text{maximize} & b^T y \\ \text{subject to} & \tilde{A}\tilde{x} = b & & \text{subject to} & \tilde{A}^T y + v + \tilde{s} = \tilde{c} \\ & \tilde{x} \in \mathcal{V} & & & v \in \mathcal{V}^\perp \\ & \tilde{x} \in \tilde{\mathcal{C}} & & & \tilde{s} \in \tilde{\mathcal{C}}^* \end{array} \tag{3.3}$$

with variables $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_l) \in \mathbf{R}^{\tilde{n}}$, $y \in \mathbf{R}^m$, $\tilde{s} = (\tilde{s}_1, \ldots, \tilde{s}_l) \in \mathbf{R}^{\tilde{n}}$, and where $\tilde{A}$ and $\tilde{c}$ are chosen to satisfy $\tilde{A}E = \sum_{k=1}^l \tilde{A}_k E_{\gamma_k} = A$ and $E^T \tilde{c} = \sum_{k=1}^l E_{\gamma_k}^T \tilde{c}_k = c$. Here $\tilde{A}_k$ and $\tilde{c}_k$ are blocks of size $|\gamma_k|$ in the partitioned matrix and vector

$$\tilde{A} = \begin{bmatrix} \tilde{A}_1 & \tilde{A}_2 & \cdots & \tilde{A}_l \end{bmatrix}, \qquad \tilde{c}^T = \begin{bmatrix} \tilde{c}_1^T & \tilde{c}_2^T & \cdots & \tilde{c}_l^T \end{bmatrix}. \tag{3.4}$$

It is straightforward to find $\tilde{A}$ and $\tilde{c}$ that satisfy these conditions. For example, one can take $\tilde{A} = AJ$, $\tilde{c} = J^T c$ with J equal to

$$J = \begin{bmatrix} P_{\gamma_1 \setminus \alpha_1} E_{\gamma_1}^T & P_{\gamma_2 \setminus \alpha_2} E_{\gamma_2}^T & \cdots & P_{\gamma_l \setminus \alpha_l} E_{\gamma_l}^T \end{bmatrix} \tag{3.5}$$

or any other left-inverse of E. However we will see later that other choices of $\tilde{A}$ may offer advantages.

The running intersection property of the intersection tree T can be used to derive a simple representation of the subspaces $\mathcal{V}$ and $\mathcal{V}^\perp$. We first note that a vector $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_l)$ is in $\mathcal{V}$ if and only if

$$E_{\alpha_j}(E_{\gamma_j}^T \tilde{x}_j - E_{\gamma_k}^T \tilde{x}_k) = 0, \quad k = 1, \ldots, l, \quad \gamma_j \in \mathrm{ch}(\gamma_k). \tag{3.6}$$

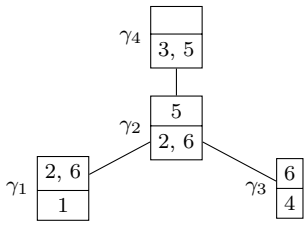

FIG. 3.1. *Spanning tree in the intersection graph of Figure 2.1.*

This can be seen as follows. Since $\alpha_j = \gamma_j \cap \gamma_{\mathrm{pa}(\gamma_j)}$, the equalities (3.6) mean that

$$E_{\gamma_j \cap \gamma_k}(E_{\gamma_j}^T \tilde{x}_j - E_{\gamma_k}^T \tilde{x}_k) = 0 \tag{3.7}$$

for all γ_k and all $\gamma_j \in \mathrm{ch}(\gamma_k)$. This is sufficient to guarantee that (3.7) holds for *all* j and k, because the running intersection property guarantees that if γ_j and γ_k intersect then their intersection is included in every index set on the path between γ_j and γ_k in the tree. The equations (3.6) therefore hold if and only if there is an x such that $\tilde{x}_k = E_{\gamma_k} x$ for $k = 1, \ldots, l$, *i.e.*, $\tilde{x} \in \mathcal{V}$. We will refer to the constraint $\tilde{x} \in \mathcal{V}$ as the *consistency constraint* in (3.3). It is needed to ensure that the variables $\tilde{x}_k$ can be interpreted as copies $\tilde{x}_k = E_{\gamma_k} x$ of overlapping subvectors of some $x \in \mathbf{R}^n$.

3.2. Correlative sparsity. The reformulated problems generalize the clique-tree conversion methods proposed for semidefinite programming in [25, 16]. These conversion methods were proposed with the purpose of reformulating large, sparse SDPs in an equivalent form that is easier to solve by interior-point methods. In this section we discuss the benefits of the reformulation in the context of general conic optimization problems with partially separable cones. The application to semidefinite programming is discussed in the next section.

The reformulated problems (3.9) are of particular interest if the sparsity of the matrix $\tilde{A}$ implies that a matrix of the form

$$\tilde{A} G \tilde{A}^T = \sum_{k=1}^{l} \tilde{A}_k G_k \tilde{A}_k^T, \tag{3.8}$$

where G is block-diagonal, with arbitrary dense diagonal blocks G_k, is sparse. We call the sparsity pattern of $\tilde{A} G \tilde{A}^T$ the *correlative sparsity* pattern of the reformulated problem (following the terminology of Kobayashi *et al.* [27]). The correlative sparsity pattern can be determined as follows: the i, j entry of $\tilde{A} G \tilde{A}^T$ is zero if there are no block columns $\tilde{A}_k$ in which the ith and jth rows are both nonzero. The correlative sparsity pattern depends on the choice of $\tilde{A}$ as illustrated by the following example.

Consider a small conic LP with $m = 4$, $n = 6$, index sets γ_k used in Figure 2.1, and a constraint matrix A with zeros in the following positions:

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 & A_{16} \\ 0 & A_{22} & 0 & 0 & A_{25} & A_{26} \\ 0 & 0 & 0 & A_{34} & 0 & A_{36} \\ 0 & 0 & A_{43} & 0 & A_{45} & 0 \end{bmatrix}.$$

In other words, equality i in $Ax = b$ involves only variables x_k for $k \in \gamma_i$. The primal

reformulated problem has a variable $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \tilde{x}_4) \in \mathbf{R}^3 \times \mathbf{R}^3 \times \mathbf{R}^2 \times \mathbf{R}^2$. If we use the intersection tree shown in Figure 3.1, the consistency constraints are

$$-\tilde{x}_{12} + \tilde{x}_{21} = 0, \qquad -\tilde{x}_{13} + \tilde{x}_{23} = 0, \qquad \tilde{x}_{23} - \tilde{x}_{32} = 0, \qquad -\tilde{x}_{22} + \tilde{x}_{42} = 0.$$

($\tilde{x}_{ij}$ denotes the jth component of $\tilde{x}_i$.) If we define $\tilde{A}$ via (3.4) and (3.5), we obtain

$$\tilde{A} = \left[\begin{array}{ccc|ccc|cc|cc} A_{11} & 0 & 0 & A_{12} & 0 & A_{16} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{22} & 0 & A_{26} & 0 & 0 & 0 & A_{25} \\ 0 & 0 & 0 & 0 & 0 & A_{36} & A_{34} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{43} & A_{45} \end{array}\right].$$

With this choice the 4×4 matrix (3.8) is dense, except for a zero in positions $(4,1)$, $(4,3)$, $(1,4)$, $(3,4)$. On the other hand, if we choose

$$\tilde{A} = \left[\begin{array}{ccc|ccc|cc|cc} A_{11} & A_{12} & A_{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & A_{22} & A_{25} & A_{26} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & A_{34} & A_{36} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & A_{43} & A_{45} \end{array}\right],$$

then the correlative sparsity pattern is diagonal.

3.3. Interior-point methods. In this section we first compare the cost of interior-point methods applied to the reformulated and the original problems, for problems with correlative sparsity.

The reformulated primal and dual problems (3.3), written in standard form, are

$$\begin{array}{lll|lll} \text{minimize} & \tilde{c}^T\tilde{x} & & \text{maximize} & b^T y & \\ \text{subject to} & \tilde{A}\tilde{x} = b, & B\tilde{x} = 0 & \text{subject to} & \tilde{A}^T y + B^T u + \tilde{s} = \tilde{c} & \\ & \tilde{x} \in \tilde{\mathcal{C}}, & & & \tilde{s} \in \tilde{\mathcal{C}}^* & \end{array} \tag{3.9}$$

where $B\tilde{x} = 0$ is the equation (3.6). An interior-point method applied to this pair of primal and dual problems requires at each iteration the solution of a linear equation (often called the *Karush-Kuhn-Tucker (KKT)* equation)

$$\left[\begin{array}{ccc} H & \tilde{A}^T & B^T \\ \tilde{A} & 0 & 0 \\ B & 0 & 0 \end{array}\right] \left[\begin{array}{c} \Delta\tilde{x} \\ \Delta y \\ \Delta u \end{array}\right] = \left[\begin{array}{c} d_{\tilde{x}} \\ d_y \\ d_u \end{array}\right] \tag{3.10}$$

where $H = \mathrm{diag}(H_1, \ldots, H_l)$ is a positive definite block-diagonal scaling matrix that depends on the algorithm used, the cones $\mathcal{C}_k$, and the current primal and dual iterates in the algorithm. Here we will assume that the blocks of H are defined as $H_k = \nabla^2\phi_k(w_k)$ where ϕ_k is a logarithmic barrier function for $\mathcal{C}_k$ and w_k is some point in $\mathbf{int}\,\mathcal{C}_k$. This assumption is sufficiently general to cover path-following methods based on primal scaling, dual scaling, and the Nesterov-Todd primal-dual scaling. In most implementations, the KKT equation is solved by eliminating $\Delta\tilde{x}$ and solving

$$\left[\begin{array}{cc} \tilde{A}H^{-1}\tilde{A}^T & \tilde{A}H^{-1}B^T \\ BH^{-1}\tilde{A}^T & BH^{-1}B^T \end{array}\right] \left[\begin{array}{c} \Delta y \\ \Delta u \end{array}\right] = \left[\begin{array}{c} \tilde{A}H^{-1}d_{\tilde{x}} - d_y \\ BH^{-1}d_{\tilde{x}} - d_u \end{array}\right]. \tag{3.11}$$

The coefficient matrix in (3.11) is called the *Schur complement matrix*. Note that the 1,1 block has the form (3.8), so its sparsity pattern is the correlative sparsity pattern.

The 2,2 block $BH^{-1}B^T$ on the other hand may be quite dense. The sparsity pattern of the coefficient matrix of (3.11) must be compared with the Schur complement matrix in an interior-point method applied to the original conic LPs (3.1). This matrix has the same sparsity pattern as the system obtained by eliminating Δu in (3.11), *i.e.*,

$$\tilde{A}(H^{-1} - H^{-1}B^T(BH^{-1}B^T)^{-1}BH^{-1})\tilde{A}^T. \tag{3.12}$$

The matrix (3.12) is often dense (due to the $BH^{-1}B^T$ term), even for problems with correlative sparsity.

An interior-point method for the reformulated problem can exploit correlative sparsity by solving (3.11) using a sparse Cholesky factorization method. If $\tilde{A}H^{-1}\tilde{A}^T$ is nonsingular, one can also explicitly eliminate Δy and reduce it to a dense linear equation in Δu with coefficient matrix

$$B(H^{-1} - H^{-1}\tilde{A}^T(\tilde{A}H^{-1}\tilde{A}^T)^{-1}\tilde{A}H^{-1})B^T.$$

To form this matrix one can take advantage of correlative sparsity. (This is the approach taken in [27].) Whichever method is used for solving (3.11), the advantage of the enhanced sparsity resulting from the sparse 1,1 block $\tilde{A}H^{-1}\tilde{A}^T$ must be weighed against the increased size of the reformulated problem. This is especially important for semidefinite programming, where the extra variables Δu are vectorized matrices, so the difference in size of the two Schur complement systems is very substantial.

3.4. Spingarn's method. Motivated by the high cost of solving the KKT equations (3.11) of the converted problem we now examine the alternative of using a first-order splitting method to exploit correlative sparsity. The converted primal problem (3.3) can be written as

$$\begin{array}{ll} \text{minimize} & f(\tilde{x}) = \tilde{c}^T\tilde{x} + \delta(\tilde{A}\tilde{x} - b) + \delta_{\tilde{\mathcal{C}}}(\tilde{x}) \\ \text{subject to} & \tilde{x} \in \mathcal{V} \end{array} \tag{3.13}$$

where δ and $\delta_{\tilde{\mathcal{C}}}$ are the indicator functions for $\{0\}$ and $\tilde{\mathcal{C}}$, respectively. Spingarn's *method of partial inverses* [37, 38] is a decomposition method for equality constrained convex optimization problems of the form (3.13). The method is known to be equivalent to the Douglas-Rachford splitting method applied to the problem of minimizing the sum $f(\tilde{x})+\delta_{\mathcal{V}}(\tilde{x})$ [15]. Starting at some $z^{(0)}$, the following three steps are repeated:

$$\tilde{x}^{(k)} = \mathrm{prox}_{f/\sigma}(z^{(k-1)}) \tag{3.14}$$

$$w^{(k)} = P_{\mathcal{V}}(2\tilde{x}^{(k)} - z^{(k-1)}) \tag{3.15}$$

$$z^{(k)} = z^{(k-1)} + \rho_k(w^{(k)} - \tilde{x}^{(k)}). \tag{3.16}$$

The algorithm depends on two parameters: a positive constant σ (we will refer to $1/\sigma$ as the *steplength*) and a *relaxation parameter* ρ_k, which can change at each iteration but must remain in an interval $(\rho_{\min}, \rho_{\max})$ with $0 < \rho_{\min} < \rho_{\max} < 2$. The operator $P_{\mathcal{V}}$ denotes Euclidean projection on $\mathcal{V}$ and $\mathrm{prox}_{f/\sigma}$ is the *proximal operator* of f, defined as

$$\mathrm{prox}_{f/\sigma}(z) = \operatorname*{argmin}_{\tilde{x}} \left(f(\tilde{x}) + \frac{\sigma}{2}\|\tilde{x} - z\|_2^2 \right).$$

It can be shown that if f is a closed convex function, then $\mathrm{prox}_{f/\sigma}(z)$ exists and is unique for all z [31, 7]. (A sufficient condition for closedness of the function f defined

in (3.13) is that there exists an $\tilde{x} \in \tilde{\mathcal{C}}$ with $\tilde{A}\tilde{x} = b$; see [36, theorem 9.3].) More details on the Douglas-Rachford method and its applications can be found in [14, 12, 7, 33].

The three steps in the Spingarn iteration can be combined in a single update

$$z^{(k)} = z^{(k-1)} - \rho_k G(z^{(k-1)}) \tag{3.17}$$

with the operator G defined as $G(z) = \text{prox}_{f/\sigma}(z) - P_{\mathcal{V}}(2\text{prox}_{f/\sigma}(z) - z)$. For $\rho_k = 1$ this is a fixed-point iteration for solving $G(z) = 0$; for other values of ρ_k it is a fixed-point iteration with relaxation (underrelaxation for $\rho_k < 1$, overrelaxation for $\rho_k > 1$). Zeros of G are related to the solutions of (3.13) as follows. If $G(z) = 0$ then $\tilde{x} = \text{prox}_{f/\sigma}(z)$ and $v = \sigma(z - \tilde{x})$ satisfy the optimality conditions for (3.13), *i.e.*,

$$\tilde{x} \in \mathcal{V}, \qquad v \in \mathcal{V}^{\perp}, \qquad v \in \partial f(\tilde{x}), \tag{3.18}$$

where $\partial f(\tilde{x})$ is the subdifferential of f at $\tilde{x}$. Conversely, if $\tilde{x}$, v satisfy these optimality conditions, then $z = \tilde{x} + (1/\sigma)v$ is a zero of G. To see this, first assume $G(z) = 0$ and define $\tilde{x} = \text{prox}_{f/\sigma}(z)$, $v = \sigma(z - \tilde{x})$. By definition of the prox-operator, $v \in \partial f(\tilde{x})$. Moreover, $G(z) = 0$ gives $\tilde{x} = P_{\mathcal{V}}(\tilde{x}) - (1/\sigma)P_{\mathcal{V}}(v)$. Therefore $\tilde{x} \in \mathcal{V}$ and $v \in \mathcal{V}^{\perp}$. Conversely suppose $\tilde{x}$, v satisfy the optimality conditions (3.18). Define $z = \tilde{x} + (1/\sigma)v$. Then it can be verified that $\tilde{x} = \text{prox}_{f/\sigma}(z)$ and $G(z) = \tilde{x} - P_{\mathcal{V}}(\tilde{x} - (1/\sigma)v) = 0$.

From step 1 in the algorithm and the definition of the proximal operator we see that the vector $v^{(k)} = \sigma(z^{(k-1)} - \tilde{x}^{(k)})$ satisfies $v^{(k)} \in \partial f(\tilde{x}^{(k)})$. If we define $r_{\text{p}}^{(k)} = P_{\mathcal{V}}(\tilde{x}^{(k)}) - \tilde{x}^{(k)}$ and $r_{\text{d}}^{(k)} = -P_{\mathcal{V}}(v^{(k)})$ then

$$\tilde{x}^{(k)} + r_{\text{p}}^{(k)} \in \mathcal{V}, \qquad v^{(k)} + r_{\text{d}}^{(k)} \in \mathcal{V}^{\perp}, \qquad v^{(k)} \in \partial f(\tilde{x}^{(k)}). \tag{3.19}$$

The vectors $r_{\text{p}}^{(k)}$ and $r_{\text{d}}^{(k)}$ can be interpreted as primal and dual residuals in the optimality conditions (3.18), evaluated at the approximate primal and dual solution $\tilde{x}^{(k)}$, $v^{(k)}$. A simple stopping criterion is therefore to terminate when

$$\frac{\|r_{\text{p}}^{(k)}\|_2}{\max\{1.0, \|\tilde{x}^{(k)}\|_2\}} \leq \epsilon_{\text{p}} \qquad \text{and} \qquad \frac{\|r_{\text{d}}^{(k)}\|_2}{\max\{1.0, \|v^{(k)}\|_2\}} \leq \epsilon_{\text{d}} \tag{3.20}$$

for some relative tolerances ϵ_{p} and ϵ_{d}.

In the standard convergence analysis of the Douglas-Rachford algorithm the parameter σ is assumed to be an arbitrary positive constant [15]. However the efficiency in practice is greatly influenced by the steplength choice and several strategies have been proposed for varying σ during the algorithm [22, 9]. As a guideline, it is often observed that the convergence is slow if one of the two residuals decreases much more rapidly than the other, and that adjusting σ can help control the balance between the primal and dual residuals. A simple strategy is to take

$$\sigma_{k+1} = \sigma_k \tau_k \text{ if } t_k > \mu, \quad \sigma_{k+1} = \sigma_k / \tau_k \text{ if } t_k < 1/\mu, \quad \sigma_{k+1} = \sigma_k \text{ otherwise,} \tag{3.21}$$

where $t_k = (\|r_{\text{p}}^{(k)}\|_2 / \|\tilde{x}^{(k)}\|_2)(\|r_{\text{d}}^{(k)}\|_2)/\|v^{(k)}\|_2)^{-1}$ is the ratio of relative primal and dual residuals, and τ_k and μ are parameters greater than one.

Projection. In our application, $\mathcal{V} = \text{Range}(E)$ with E defined in (3.2), and the Euclidean projection $P_{\mathcal{V}}(\tilde{x})$ is easy to compute. For each $i \in \{1, 2, \ldots, n\}$, define $M(i) = \{k \mid i \in \gamma_k\}$. The vertices of T indexed by $M(i)$ define a subtree (this is a

consequence of the running intersection property). The projection of $\tilde{x}$ on $\mathcal{V}$ is the vector $P_{\mathcal{V}}(\tilde{x}) = (E_{\gamma_1}(\bar{x}), E_{\gamma_2}(\bar{x}), \ldots, E_{\gamma_l}(\bar{x}))$ where $\bar{x}$ is the n-vector with components

$$\bar{x}_i = \frac{(\sum_{k \in M(i)} E_{\gamma_k}^T \tilde{x}_k)_i}{|M(i)|}, \quad i = 1 \ldots, n.$$

In other words, component i of $\bar{x}$ is a simple average of the corresponding components of $\tilde{x}_k$, for the sets γ_k that contain i.

Proximal operator. The value $\tilde{x} = \text{prox}_{f/\sigma}(z)$ of the prox-operator of the function f in (3.13) is the primal solution in the pair of conic quadratic optimization problems (conic QPs)

$$\begin{array}{lll} \text{min.} & \tilde{c}^T\tilde{x} + \dfrac{\sigma}{2}\|\tilde{x} - z\|_2^2 & \text{max.} \quad b^T y - \dfrac{1}{2\sigma}\|\tilde{c} - \tilde{A}^T y - \sigma z - \tilde{s}\|_2^2 \\ \text{s.t.} & \tilde{A}\tilde{x} = b, \quad \tilde{x} \in \tilde{\mathcal{C}} & \text{s.t.} \quad \tilde{s} \in \tilde{\mathcal{C}}^* \end{array} \tag{3.22}$$

with primal variables $\tilde{x}$ and dual variables y, $\tilde{s}$. Equivalently, $\tilde{x}$, y, $\tilde{s}$ satisfy

$$\tilde{A}\tilde{x} = b, \qquad \tilde{A}^T y + \tilde{s} + \sigma(z - \tilde{x}) = \tilde{c}, \qquad \tilde{x} \in \tilde{\mathcal{C}}, \qquad \tilde{s} \in \tilde{\mathcal{C}}^*, \qquad \tilde{x}^T\tilde{s} = 0. \tag{3.23}$$

We will assume that the prox-operator of f is computed exactly, *i.e.*, we do not explore the possibility of speeding up the algorithm by using inexact prox-evaluations. This is justified if an interior-point method is used for solving (3.22), since interior-point methods achieve a high accuracy and offer only a modest gain in efficiency if solutions with low accuracy are acceptable. Using the optimality conditions (3.23) that characterize $\tilde{x}^{(k)} = \text{prox}_{f/\sigma}(z^{(k-1)})$ we can then be more specific about the accuracy of $\tilde{x}^{(k)}$ as an approximate solution of the conic LPs (3.3). By solving the primal and dual conic QPs we find $\tilde{x}^{(k)}$, $y^{(k)}$, $\tilde{s}^{(k)}$ that satisfy $\tilde{x}^{(k)} \in \tilde{\mathcal{C}}$, $\tilde{s}^{(k)} \in \tilde{\mathcal{C}}^*$, and

$$\tilde{A}\tilde{x}^{(k)} = b, \qquad \tilde{A}^T y^{(k)} + \tilde{s}^{(k)} + v^{(k)} = \tilde{c}, \qquad (\tilde{x}^{(k)})^T\tilde{s}^{(k)} = 0 \tag{3.24}$$

where $v^{(k)} = \sigma(z^{(k-1)} - \tilde{x}^{(k)})$. These are exactly the primal-dual optimality conditions for the conic LPs (3.3), except for the constraints involving $\mathcal{V}$ and $\mathcal{V}^\perp$. The primal and dual residuals $r_{\text{p}}^{(k)} = P_{\mathcal{V}}(\tilde{x}^{(k)}) - \tilde{x}^{(k)}$ and $r_{\text{d}}^{(k)} = -P_{\mathcal{V}}(v^{(k)})$ measure the deviation of $\tilde{x}^{(k)}$ from $\mathcal{V}$ and of $\tilde{v}^{(k)}$ from $\mathcal{V}^\perp$.

We now comment on the cost of evaluating the prox-operator by solving the conic QPs (3.22). An interior-point method applied to this problem requires the solution of KKT systems of the form

$$\begin{bmatrix} \sigma I + H & \tilde{A}^T \\ \tilde{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta y \end{bmatrix} = \begin{bmatrix} d_{\tilde{x}} \\ d_y \end{bmatrix}$$

where H is a block-diagonal positive definite scaling matrix. As before, we assume that the diagonal blocks of H are of the form $H_k = \nabla^2\phi_k(w_k)$ where ϕ_k is a logarithmic barrier of $\mathcal{C}_k$. The cost per iteration of evaluating the proximal operator is dominated by the cost of assembling the coefficient matrix

$$\tilde{A}(\sigma I + H)^{-1}\tilde{A}^T = \sum_{k=1}^{l} \tilde{A}_k(\sigma I + H_k)^{-1}\tilde{A}_k^T \tag{3.25}$$

in the Schur complement equation $\tilde{A}(\sigma I + H)^{-1}\tilde{A}^T\Delta y = \tilde{A}(\sigma I + H)^{-1}d_x - d_y$ and the cost of solving the Schur complement system. For many types of conic LPs the extra

term σI in (3.25) can be handled by simple changes in the interior-point algorithm. This is true in particular when H_k is diagonal or diagonal-plus-low-rank, as is the case when $\mathcal{C}_k$ is a nonnegative orthant or second-order cone. For positive semidefinite cones the modifications are more involved and will be discussed in section 5.2. In general, it is therefore fair to assume that in most applications the cost of assembling the Schur complement matrix in (3.25) is roughly the same as the cost of computing $\tilde{A}^T H^{-1} \tilde{A}^T$. Since the Schur complement matrix in (3.25) is sparse (under the assumption of correlative sparsity), it can be factored at a smaller cost than its counterpart (3.11) for the reformulated conic LPs. Depending on the level of correlative sparsity, one evaluation of the proximal operator via an interior-point method can be substantially less expensive than solving the reformulated problems by an interior-point method.

4. Sparse semidefinite optimization. In the rest of the paper we discuss the application to sparse semidefinite optimization. We first explain why sparse SDPs with a chordal sparsity pattern can be viewed as examples of partially separable structure. In section 5 we then apply the decomposition method described in section 3.4.

We formally define a symmetric sparsity pattern of order p as a set of index pairs $V \subseteq \{1, 2, \ldots, p\} \times \{1, 2, \ldots, p\}$ with the property that $(i, j) \in V$ whenever $(j, i) \in V$. We say a symmetric matrix X of order p has sparsity pattern V if $X_{ij} = 0$ when $(i, j) \notin V$. The entries X_{ij} for $(i, j) \in V$ are referred to as the *nonzero entries* of X, even though they may be numerically zero. The set of symmetric matrices of order p with sparsity pattern V is denoted $\mathbf{S}^p_V$.

4.1. Nonsymmetric formulation. Consider a semidefinite program (SDP) in the standard form and its dual:

$$
\begin{array}{lll}
\text{min.} & \mathbf{tr}(CX) & \\
\text{s.t.} & \mathbf{tr}(F_i X) = b_i, \quad i = 1, \ldots, m & \\
& X \succeq 0 &
\end{array}
\qquad\qquad
\begin{array}{lll}
\text{max.} & b^T y & \\
\text{s.t.} & \sum_{i=1}^m y_i F_i + S = C & \\
& S \succeq 0. &
\end{array}
\tag{4.1}
$$

The primal variable is a symmetric matrix $X \in \mathbf{S}^p$; the dual variables are $y \in \mathbf{R}^m$ and the slack matrix $S \in \mathbf{S}^p$. The problem data are the vector $b \in \mathbf{R}^m$ and the matrices C, $F_i \in \mathbf{S}^p$. The *aggregate* sparsity pattern is the union of the sparsity patterns of C, F_1, …, F_m. If V is the aggregate sparsity pattern, then we can take C, $F_i \in \mathbf{S}^p_V$. The dual variable S is then necessarily sparse at any dual feasible point, with the same sparsity pattern V. The primal variable X, on the other hand, is dense in general, but one can note that the cost function and the equality constraints only depend on the entries of X in the positions of the nonzeros of the sparsity pattern V. The other entries of X are arbitrary, as long as the matrix is positive semidefinite. The primal and dual problems can therefore be viewed alternatively as conic linear optimization problems with respect to a pair of non-self-dual cones:

$$
\begin{array}{lll}
\text{min.} & \mathbf{tr}(CX) & \\
\text{s.t.} & \mathbf{tr}(F_i X) = b_i, \quad i = 1, \ldots, m & \\
& X \in \mathbf{S}^p_{V,\mathrm{c}} &
\end{array}
\qquad\qquad
\begin{array}{lll}
\text{max.} & b^T y & \\
\text{s.t.} & \sum_{i=1}^m y_i F_i + S = C & \\
& S \in \mathbf{S}^p_{V,+}. &
\end{array}
\tag{4.2}
$$

Here the variables X and S, as well as the coefficient matrices C, F_i, are matrices in $\mathbf{S}^p_V$. The primal cone $\mathbf{S}^p_{V,\mathrm{c}}$ is the set of matrices in $\mathbf{S}^p_V$ that have a positive semidefinite completion, *i.e.*, the projection of the cone of positive semidefinite matrices of order p on the subspace $\mathbf{S}^p_V$. We will refer to $\mathbf{S}^p_{V,\mathrm{c}}$ as the sparse *p.s.d.-completable cone*. The

dual cone $\mathbf{S}^p_{V,+}$ is the set of positive semidefinite matrices in $\mathbf{S}^p_V$, *i.e.*, the intersection of the cone of positive semidefinite matrices of order p with the subspace $\mathbf{S}^p_V$. This cone will be referred to as the sparse p.s.d. cone. It can be shown that the two cones form a dual pair of proper convex cones, provided the nonzero positions in the sparsity pattern V include the diagonal entries.

It is often convenient to use vector notation for the matrix variables in (4.2). For this purpose we introduce an operator $x = \mathbf{vec}_V(X)$ that maps the lower-triangular nonzeros of a matrix $X \in \mathbf{S}^p_V$ to a vector x of length $n = (|V| + p)/2$, using a format that preserves inner products, *i.e.*, $\mathbf{tr}(XY) = \mathbf{vec}_V(X)^T\mathbf{vec}_V(Y)$ for all X, Y. For example, one can copy the nonzero lower-triangular entries of X in column-major order to x, scaling the strictly lower-triangular entries by $\sqrt{2}$. A similar notation $x = \mathbf{vec}(X)$ (without subscript) will be used for a packed vector representation of a dense matrix: if $X \in \mathbf{S}^p$, then $x = \mathbf{vec}(X)$ is a vector of length $p(p+1)/2$ containing the lower-triangular entries of X in a storage format that preserves the inner products. Using this notation, the matrix cones $\mathbf{S}^p_{V,\mathrm{c}}$ and $\mathbf{S}^p_{V,+}$ can be 'vectorized' to define two cones $\mathcal{C} = \{\mathbf{vec}_V(X) \mid X \in \mathbf{S}^p_{V,\mathrm{c}}\}$ and $\mathcal{C}^* = \{\mathbf{vec}_V(S) \mid S \in \mathbf{S}^p_{V,+}\}$. These cones form a dual pair of proper convex cones in $\mathbf{R}^n$ with $n = (|V| + p)/2$. The conic linear optimization problems (4.2) can then be written as (3.1) with variables $x = \mathbf{vec}_V(X)$, $s = \mathbf{vec}_V(S)$, y, and problem parameters $c = \mathbf{vec}_V(C)$, $A = \left[\begin{array}{cccc} \mathbf{vec}_V(F_1) & \mathbf{vec}_V(F_2) & \cdots & \mathbf{vec}_V(F_m) \end{array}\right]^T$.

4.2. Clique decomposition of chordal sparse matrix cones. The nonsymmetric conic optimization or matrix completion approach to sparse semidefinite programming, based on the formulation (4.2), was proposed by Fukuda *et al.* [16] and further developed in [32, 10, 39, 4, 25]. The various techniques described in these papers assume that the sparsity pattern V is chordal. In this section we review some key results concerning positive semidefinite matrices with chordal sparsity.

With each sparsity pattern V one associates an undirected graph $\mathcal{G}_V$ with p vertices and edges $\{i, j\}$ between pairs of vertices $(i, j) \in V$ with $i > j$. A clique in $\mathcal{G}_V$ is a maximal complete subgraph, *i.e.*, a maximal set $\beta \subseteq \{1, 2, \ldots, p\}$ such that $\beta \times \beta \subseteq V$. Each clique defines a maximal dense principal submatrix in any matrix with sparsity pattern V. If the cliques in the graph $\mathcal{G}_V$ are β_k, $k = 1, \ldots, l$, then the sparsity pattern V can be expressed as $V = \bigcup_{k=1,\ldots,l} \beta_k \times \beta_k$. A sparsity pattern V is called chordal if the graph $\mathcal{G}_V$ is chordal.

In the remainder of the paper we assume that V is a chordal sparsity pattern that contains all the diagonal entries ($(i, i) \in V$ for $i = 1, \ldots, p$). We denote by β_k, $k = 1, \ldots, l$, the cliques of $\mathcal{G}_V$ and define $V_k = \beta_k \times \beta_k$. We will use two classical theorems that characterize the cones $\mathbf{S}^p_{V,\mathrm{c}}$ and $\mathbf{S}^p_{V,+}$ for chordal patterns V. The first theorem [20, theorem 4] [1, theorem 2.3] [24, theorem 1] states that the sparse p.s.d. cone $\mathbf{S}^p_{V,+}$ is a sum of positive semidefinite cones with simple sparsity patterns:

$$\mathbf{S}^p_{V,+} = \sum_{k=1}^{l} \mathbf{S}^p_{V_k,+} = \{\sum_{k=1}^{l} \mathcal{E}^*_{\beta_k}(\tilde{S}_k) \mid \tilde{S}_k \in \mathbf{S}^{|\beta_k|}_+\} \tag{4.3}$$

where $\mathbf{S}^{|\beta_k|}_+$ is the positive semidefinite cone of order $|\beta_k|$. The operator $\mathcal{E}^*_{\beta_k}$ copies a dense matrix of order $|\beta_k|$ to the principal submatrix indexed by β_k in a symmetric matrix of order p; see section 1. According to the decomposition result (4.3), every positive semidefinite matrix X with sparsity pattern V can be decomposed as a sum of positive semidefinite matrices, each with a sparsity pattern consisting of a single

principal dense block $V_k = \beta_k \times \beta_k$. If X is positive definite, a decomposition of this form is easily calculated via a zero-fill Cholesky factorization.

The second theorem characterizes the p.s.d.-completable cone $\mathbf{S}^p_{V,\mathrm{c}}$ [21, theorem 7]:

$$\mathbf{S}^p_{V,\mathrm{c}} = \{X \in \mathbf{S}^p_V \mid \mathcal{E}_{\beta_k}(X) \in \mathbf{S}^{|\beta_k|}_+,\ k = 1, \ldots, l\}. \tag{4.4}$$

The operator $\mathcal{E}_{\beta_k}$ extracts from its argument the dense principal submatrix indexed by β_k. (This is the adjoint operation of $\mathcal{E}^*_{\beta_k}$; see section 1.) In other words, a matrix in $\mathbf{S}^p_V$ has a positive semidefinite completion if and only if all its maximal dense principal submatrices $X_{\beta_k\beta_k}$ are positive semidefinite. This result can be derived from (4.3) and the duality of the cones $\mathbf{S}^p_{V,\mathrm{c}}$ and $\mathbf{S}^p_{V,+}$; see [24, corollary 2].

We now express the clique decomposition formulas (4.3) and (4.4) in vector notation. For each clique β_k, define an index set $\gamma_k \subseteq \{1, 2, \ldots, n\}$ via the identity

$$E_{\gamma_k}\mathbf{vec}_V(Z) = \mathbf{vec}(Z_{\beta_k\beta_k}) \quad \forall Z \in \mathbf{S}^p_V. \tag{4.5}$$

The index set γ_k has length $|\gamma_k| = |\beta_k|(|\beta_k| + 1)/2$ and its elements indicate the positions of the entries of the $\beta_k \times \beta_k$ submatrix of Z in the vectorized matrix $\mathbf{vec}_V(Z)$. Using this notation, the cone $\mathcal{C}$ can be expressed as (2.1) where $\mathcal{C}_k = \{\mathbf{vec}(W) \mid W \in \mathbf{S}^{|\beta_k|}_+\}$ is the vectorized dense positive semidefinite matrix cone of order $|\beta_k|$. The clique decomposition (4.4) of the p.s.d. cone can be expressed in vector notation as (2.2). (Note that $\mathcal{C}_k$ is self-dual, so here $\mathcal{C}_k = \mathcal{C}^*_k$.) The decomposition result (4.4) shows that the p.s.d.-completable cone associated with a chordal sparsity pattern V is partially separable. We also note that the assumptions in section 2 are satisfied for this choice of index sets γ_k and cones $\mathcal{C}_k$. (The third assumption holds with $\bar{x} = \mathbf{vec}_V(I)$ because the sparsity pattern V includes the diagonal entries.)

The cliques β_k of V can be arranged in a clique tree that satisfies the running intersection property ($\beta_i \cap \beta_j \subseteq \beta_k$ if clique k is on the path between cliques β_i and β_j in the tree); see [8]. We denote by η_k the intersection of the clique β_k with its parent in the clique tree. Since there is a one-to-one relation between the index sets γ_k defined in (4.5) and the cliques β_k of $\mathcal{G}_V$, we can identify the clique graph of $\mathcal{G}_V$ (which has vertices β_k) with the intersection graph for the index sets γ_k. Similarly, we do not have to distinguish between a clique tree T for $\mathcal{G}_V$ and a spanning tree with the running intersection property in the intersection graph of the sets γ_k. The sets $\alpha_k = \gamma_k \cap \mathrm{pa}(\gamma_k)$ are in a one-to-one relation to the sets $\eta_k = \beta_k \cap \mathrm{pa}(\beta_k)$ via the identity $E_{\alpha_k}(\mathbf{vec}_V(Z)) = \mathbf{vec}(Z_{\eta_k\eta_k})$ for arbitrary $Z \in \mathbf{S}^p_V$.

Figure 4.1 illustrates this notation. The cliques are $\beta_1 = \{1, 2, 3\}$, $\beta_2 = \{2, 3, 4\}$, $\beta_3 = \{3, 4, 5\}$. If we use the column-major order for the nonzero entries in the vectorized matrix, the cliques correspond to the index sets $\gamma_1 = \{1, 2, 3, 4, 5, 7\}$, $\gamma_2 = \{4, 5, 6, 7, 8, 10\}$, $\gamma_3 = \{7, 8, 9, 10, 11, 12\}$. The sets $\eta_k = \beta_k \cap \mathrm{pa}(\beta_k)$ and $\alpha_k = \gamma_k \cap \mathrm{pa}(\gamma_k)$ are $\eta_1 = \{2, 3\}$, $\eta_2 = \{3, 4\}$, $\eta_3 = \{\}$, $\alpha_1 = \{4, 5, 7\}$, $\alpha_2 = \{7, 8, 10\}$, $\alpha_3 = \{\}$.

5. Decomposition in semidefinite programming. We now work out the details of the decomposition method when applied to sparse semidefinite programming. In particular, we describe an efficient method for solving the quadratic conic optimization problem (3.22), needed for the evaluation of the proximal operator, when the cone $\tilde{\mathcal{C}}$ is a product of positive semidefinite matrix cones.

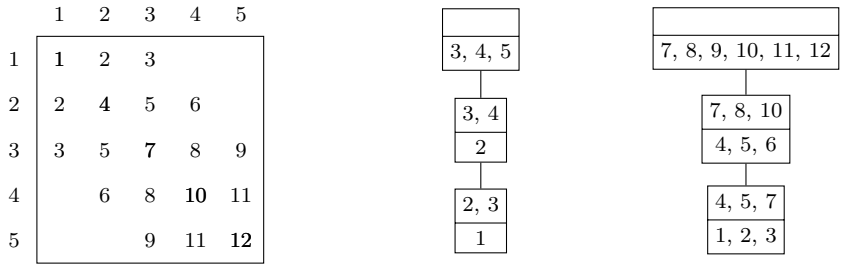

FIG. 4.1. *A 5×5 chordal pattern with 12 nonzero entries in the lower triangular part. The numbers in the matrix are the indices of the entries in the vectorized matrix. The center shows a clique tree. The right-hand part shows the corresponding spanning tree in the intersection graph.*

5.1. Converted problems. We first express the reformulated problems (3.3) and (3.9) for SDPs in matrix notation. The reformulated primal problem is

$$
\begin{array}{lll}
\text{minimize} & \displaystyle\sum_{k=1}^{l} \mathbf{tr}(\tilde{C}_k \tilde{X}_k) & \\
\text{subject to} & \displaystyle\sum_{k=1}^{l} \mathbf{tr}(\tilde{F}_{ik} \tilde{X}_k) = b_i, \quad i = 1, \ldots, m & \\
& \mathcal{E}_{\eta_j}(\mathcal{E}^*_{\beta_k}(\tilde{X}_k) - \mathcal{E}^*_{\beta_j}(\tilde{X}_j)) = 0, \quad k = 1, \ldots, l, \quad \beta_j \in \mathrm{ch}(\beta_k) & \\
& \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l &
\end{array}
\tag{5.1}
$$

with variables $\tilde{X}_k \in \mathbf{S}^{|\beta_k|}$, $k = 1, \ldots, l$. The coefficient matrices $\tilde{C}_k$ and $\tilde{F}_{ik}$ are chosen so that $\mathbf{tr}(CZ) = \sum_{k=1}^{l} \mathbf{tr}(\tilde{C}_k Z_{\beta_k \beta_k})$ and $\mathbf{tr}(F_i Z) = \sum_{k=1}^{l} \mathbf{tr}(\tilde{F}_{ik} Z_{\beta_k \beta_k})$ for all $Z \in \mathbf{S}^p_V$. One possible choice is $\tilde{C}_k = \mathcal{E}_{\beta_k}(C - \mathcal{P}_{\eta_k}(C))$ and $\tilde{F}_{ik} = \mathcal{E}_{\beta_k}(F_i - \mathcal{P}_{\eta_k}(F_i))$. The variables $\tilde{X}_k$ in (5.1) are interpreted as copies of the dense submatrices $X_{\beta_k \beta_k}$. The second set of equality constraints in (5.1) are the consistency constraints that ensure that the entries of $\tilde{X}_k$ agree when they refer to the same entry of X.

The converted dual problem is

$$
\begin{array}{lll}
\text{max.} & b^T y & \\
\text{s.t.} & \displaystyle\sum_{i=1}^{m} y_i \tilde{F}_{ik} + \mathcal{E}_{\beta_k}(\mathcal{E}^*_{\eta_k}(U_k) - \sum_{\beta_j \in \mathrm{ch}(\beta_k)} \mathcal{E}^*_{\eta_j}(U_j)) + \tilde{S}_k = \tilde{C}_k, \quad k = 1, \ldots, l & \\
& \tilde{S}_k \succeq 0, \quad k = 1, \ldots, l, &
\end{array}
\tag{5.2}
$$

with variables y, $\tilde{S}_k \in \mathbf{S}^{|\beta_k|}$, and $U_k \in \mathbf{S}^{|\eta_k|}$, $k = 1, \ldots, l$. The reformulations (5.1) and (5.2) also follow from the clique-tree conversion methods proposed in [25, 16].

5.2. Proximal operator. In the clique-tree conversion methods of [32, 25] the converted SDP (5.1) is solved by an interior-point method. A limitation to this approach is the large number of equality constraints added in the primal problem or, equivalently, the large dimension of the auxiliary variables U_k in the dual problem. In section 3.4 we proposed an operator-splitting method to address this problem. The key step in each iteration of the splitting method is the evaluation of a proximal

operator, by solving the quadratic conic optimization problem (QP)

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{l} \mathbf{tr}(\tilde{C}_k \tilde{X}_k) + (\sigma/2) \sum_{k=1}^{l} \|\tilde{X}_k - Z_k\|_F^2 \\ \text{subject to} & \sum_{k=1}^{l} \mathbf{tr}(\tilde{F}_{ik} \tilde{X}_k) = b_i, \quad i = 1, \ldots, m \\ & \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l. \end{array} \tag{5.3}$$

Solving this problem by a general-purpose solver can be quite expensive and most solvers require a reformulation to remove the quadratic term in the objective by adding second-order cone constraints. However the problem can be solved efficiently via a customized interior-point solver, as we now describe. A similar technique was used for handling variable bounds in SDPs in [3, 41].

The Newton equation or KKT system that must be solved in each iteration of an interior-point method for the conic QP (5.3) has the form

$$\sigma \Delta \tilde{X}_k + W_k \Delta \tilde{X}_k W_k + \sum_{i=1}^{m} \Delta y_i \tilde{F}_{ik} = D_k, \quad k = 1, \ldots, l \tag{5.4}$$

$$\sum_{k=1}^{l} \mathbf{tr}(\tilde{F}_{ik}\, \Delta \tilde{X}_k) = d_i, \quad i = 1, \ldots, m, \tag{5.5}$$

with variables $\Delta \tilde{X}_k$, Δy, where W_k is a positive definite scaling matrix. The first term $\sigma \Delta \tilde{X}_k$ results from the quadratic term in the objective. Without this term it is straightforward to eliminate the variable $\Delta \tilde{X}_k$ from the first equation, to obtain an equation in the variable Δy. To achieve the same goal at a similar cost with a customized solver we first compute eigenvalue decompositions $W_k = Q_k \operatorname{diag}(\lambda_k) Q_k^T$ of the scaling matrices, and define l matrices $\Gamma_k \in \mathbf{S}^{|\beta_k|}$ with entries $(\Gamma_k)_{ij} = 1/(\sigma + \lambda_{ki}\lambda_{kj})$ for $i, j = 1, \ldots, |\beta_k|$. We can now use (5.4) to express $\Delta \tilde{X}_k$ in terms of Δy as $\Delta \tilde{X}_k = Q_k \left(\Gamma_k \circ (\widehat{D}_k - \sum_{i=1}^{m} \Delta y_i \widehat{F}_{ik})\right) Q_k^T$ with $\widehat{D}_k = Q_k^T D_k Q_k$, $\widehat{F}_{ik} = Q_k^T \tilde{F}_{ik} Q_k$, and where $\circ$ denotes the Hadamard (component-wise) product. Substituting the expression for $\Delta \tilde{X}_k$ in (5.5) gives an equation $H \Delta y = g$ of order m, with

$$H_{ij} = \sum_{k=1}^{l} \mathbf{tr}(\widehat{F}_{ik}(\Gamma_k \circ \widehat{F}_{jk})), \qquad g_i = -d_i + \sum_{k=1}^{l} \mathbf{tr}(\widehat{F}_{ik}(\Gamma_k \circ D_k)). \tag{5.6}$$

The cost of this solution method for the KKT system (5.4)–(5.5) is comparable to the cost of solving the KKT systems in an interior-point method applied to the conic optimization problem (5.3) without the quadratic term. The proximal operator can therefore be evaluated at roughly the same cost as the cost of solving the converted SDP (5.1) with the consistency constraints removed.

To illustrate the value of this technique, we compare in Figure 5.1 the time needed to solve the semidefinite QP (5.3) using several methods: the general-purpose conic solver SDPT3 for MATLAB, called directly or via CVX (version 2.0 beta) [18, 17], and an implementation of the algorithm described above in CVXOPT [2, 3]. The problems are dense and randomly generated with $l = 1$, $m = p = |\beta_1|$, and $\sigma = 1$. The figure shows CPU time versus the order p of the matrix variable, computed on an Intel Xeon CPU E3-1225 processor with 4 cores, 3.10 GHz clock speed, and 32 GB RAM. For the fast prox implementation, CPU time is measured using the Python

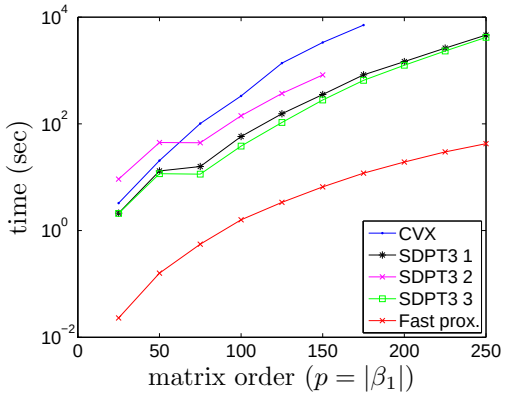

FIG. 5.1. *Time required for a single proximal operator evaluation (5.3) on a dense subproblem with one clique ($l = 1$) of size $p = |\beta_1|$ and $m = p$ constraints (averaged over 10 trials). The CPU time of the general-purpose solver SDPT3, called directly or via CVX, is compared against a customized fast proximal operator.*

module psutils (available at `code.google.com/p/psutil`). For CVX and SDPT3, we used the CPU times reported by the solver.

The fast algorithm uses the cone QP solver in CVXOPT with the default termination criteria. In the CVX code the function `sum_square()` was used to represent the quadratic term in the objective. The three SDPT3 curves correspond to different ways of converting the problem to a conic LP using the equivalence

$$u^T u \leq v \qquad \Longleftrightarrow \qquad \left\| \begin{bmatrix} 2u \\ v-1 \end{bmatrix} \right\|_2 \leq v+1.$$

In the first formulation ('SDPT3 1') we replace the squared norm in the objective with a variable w and add the constraint $\|\tilde{X} - Z\|_F^2 \leq w$ via a single second-order cone constraint of order $p(p+1)/2 + 1$. In the second formulation ('SDPT3 2'), we replace the quadratic term with a sum of $p(p+1)/2$ auxiliary variables w_{ij}, for $1 \leq i \leq j \leq p$, and add $p(p+1)/2$ second-order cone constraints of dimension two to express $(\tilde{X}_{ij} - Z_{ij})^2 \leq w_{ij}$. In the third formulation ('SDPT3 3') we replace the quadratic term with the sum of p variables w_i, subject to the constraint that w_i is an upper bound on the Euclidean norm of the lower-triangular part of the ith column of $\tilde{X} - Z$. As can be seen, the choice between the conic LP reformulations has an effect on the efficiency of a general-purpose solver. However the experiment also confirms that the fast proximal operator evaluation is orders of magnitude faster than general-purpose solvers applied to equivalent conic LPs.

5.3. Correlative sparsity. The efficiency of the decomposition method depends crucially on the cost of the proximal operator evaluations, which is determined by the sparsity pattern of the Schur complement matrix H (5.6), *i.e.*, the correlative sparsity pattern of the reformulated problems. Note that in general the scaled matrices $\widehat{F}_{ik}$ used to assemble H will be either completely dense (if $\tilde{F}_{ik} \neq 0$) or zero (if $\tilde{F}_{ik} = 0$). Therefore $H_{ij} = 0$ if for each k at least one of the coefficient matrices $\tilde{F}_{ik}$ and $\tilde{F}_{jk}$ is zero. This rule characterizes the correlative sparsity pattern.

As pointed out in section 3.2, the correlative sparsity can be enhanced by exploiting the flexibility in the choice of parameters of the reformulated problem (the

matrices $\tilde{F}_{ik}$). While the optimal choice is not clear in general, it is straightforward in the important special case when the index set $\{1, \ldots, m\}$ can be partitioned in l sets $\nu_1, \ldots, \nu_l$, with the property that if $i \in \nu_j$, then all the nonzero entries of F_i belong to the principal submatrix $(F_i)_{\beta_j\beta_j}$. In other words $F_i = \mathcal{P}_{\beta_j}(F_i)$ for $i \in \nu_j$. In this case, a valid choice for the coefficient matrices $\tilde{F}_{ik}$ is to take $\tilde{F}_{ij} = \mathcal{E}_{\beta_j}(F_i)$ and $\tilde{F}_{ik} = 0$ $(k \neq j)$ when $i \in \nu_j$. With this choice, the matrix H can be re-ordered to be block-diagonal with dense blocks $H_{\nu_i\nu_i}$. Moreover the QP (5.3) is separable and equivalent to l independent problems

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(\tilde{C}_k\tilde{X}_k) + (\sigma/2)\|\tilde{X}_k - Z_k\|_F^2 \\ \text{subject to} & \mathbf{tr}(\tilde{F}_{ik}\tilde{X}_k) = b_i, \quad i \in \nu_k \\ & \tilde{X}_k \succeq 0. \end{array}$$

6. Numerical examples. In this section we present the results of numerical experiments with the decomposition method applied to semidefinite programs. First, we apply the decomposition method to an approximate Euclidean distance matrix completion problem, motivated by an application in sensor network node localization, and illustrate the convergence behavior of the method in practice. The problem involves a sparse matrix variable whose sparsity pattern is characterized by the sensor network topology, and is interesting because in the converted form the problem has block-diagonal correlative sparsity regardless of the network topology. In the second experiment, we present results for a family of problems with block-arrow aggregate sparsity and block-diagonal correlative sparsity. By comparing the CPU times required by general-purpose interior-point methods and the decomposition method, we are able to characterize the regime in which each method is more efficient.

The decomposition method is implemented in Python (version 2.6.5), using the conic QP solver of CVXOPT (version 1.1.5) for solving the conic QPs (5.3) in the evaluation of the proximal operators. SEDUMI (version 1.1) [40] and SDPT3 (version 4.0) in MATLAB (version 2011b) are used as the general-purpose solver for the experiments in sections 6.2 and 6.1. The experiments are performed on an Intel Xeon CPU E3-1225 processor (4 cores, 3.10 GHz clock) and 8 GB RAM, running Ubuntu.

In Spingarn's method we use the stopping condition (3.20). The terms 'relative primal and dual residuals' will refer to the left-hand sides in these inequalities. As argued in section 3.4 these residuals determine the error after k iterations of the Spingarn algorithm, if we assume the error in the prox-operators is negligible.

6.1. Approximate Euclidean distance matrix completion. A Euclidean distance matrix (EDM) D is a matrix with entries that can be expressed as squared pairwise distances $D_{ij} = \|x_i - x_j\|_2^2$ for some set of vectors x_k. In this section, we consider the problem of fitting a Euclidean distance matrix to measurements $\widehat{D}_{ij}$ of a subset of its entries. This and related problems arise in many applications, including, for example, the sensor network node localization problem [11, 26].

Expanding the identity in the definition of Euclidean distance matrix, $D_{ij} = x_i^T x_i - 2x_i^T x_j + x_j^T x_j$, shows that a matrix D is a Euclidean distance matrix if and only if $D_{ij} = X_{ii} - 2X_{ij} + X_{jj}$ for a positive semidefinite matrix X (the Gram matrix with entries $X_{ij} = x_i^T x_j$). Furthermore, since D only depends on the pairwise distances of the configuration points, we can arbitrarily place one of the points at the origin or, equivalently, set one row and column of X to zero. This gives an equivalent characterization: D is a $(p+1) \times (p+1)$ Euclidean distance matrix if and only if there exists a positive semidefinite matrix $X \in \mathbf{S}^p$ such that $D_{ij} = \mathbf{tr}(F_{ij}X)$ for

$1 \le i < j \le p+1$ where $F_{ij} = (e_i - e_j)(e_i - e_j)^T$ for $1 \le i < j \le p$ and $F_{ij} = e_i e_i^T$ for $1 \le i < j = p+1$. Here e_i denotes the ith unit vector in $\mathbf{R}^p$.

In the EDM approximation problem we are given a set of measurements $\widehat{D}_{ij}$ for entries $(i,j) \in W$ where $W \subseteq \{(i,j) \mid 1 \le i < j \le p+1\}$. The problem of fitting a Euclidean distance matrix to the measurements can be posed as

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(i,j)\in W} |\,\mathbf{tr}(F_{ij}X) - \widehat{D}_{ij}| \\ \text{subject to} & X \succeq 0, \end{array} \tag{6.1}$$

with variable $X \in \mathbf{S}^p$. (We choose the ℓ_1-norm to measure the quality of the fit simply because the problem is more easily expressed as a conic LP.) Now let V be a chordal sparsity pattern of order p that includes the aggregate sparsity pattern of the matrices F_{ij}. In other words, if $(i,j) \in W$ with $1 \le i < j \le p$, then (i,j) is a nonzero in V. Moreover V is chordal and includes all the diagonal entries in its nonzeros. Such a pattern V is called a *chordal embedding* of W. Then, without loss of generality, we can restrict the variable X to be a matrix in $\mathbf{S}^p_V$ and we obtain the equivalent problem

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(i,j)\in W} |\,\mathbf{tr}(F_{ij}X) - \widehat{D}_{ij}| \\ \text{subject to} & X \in \mathbf{S}^p_{V,\mathrm{c}}. \end{array} \tag{6.2}$$

This problem is readily converted into a standard conic LP of the form (4.2), which can then be solved using the method of section 5. An interesting feature of this application is that the correlative sparsity in the converted problem is block-diagonal.

The conversion method and the block-diagonal correlative sparsity can also be explained directly in terms of the problem (6.2). Suppose V has l cliques β_k, $k = 1, \ldots, l$. Suppose we partition the set W into l sets W_k with the property that if $(i,j) \in W_k$ and $1 \le i < j \le p$, then $i, j \in \beta_k$, and if $(i, p+1) \in W_k$, then $i \in \beta_k$. Then (6.2) is equivalent to

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{k=1}^{l} \sum_{(i,j)\in W_k} |\,\mathbf{tr}(F_{ij}\mathcal{E}^*_{\beta_k}(\tilde{X}_k)) - \widehat{D}_{ij}| \\ \text{subject to} & \mathcal{E}_{\eta_j}(\mathcal{E}^*_{\beta_k}(\tilde{X}_k) - \mathcal{E}^*_{\beta_j}(\tilde{X}_j)) = 0, \quad k = 1, \ldots, l, \quad \beta_j \in \mathrm{ch}(\beta_k) \\ & \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l, \end{array} \tag{6.3}$$

with variables $\tilde{X}_k \in \mathbf{S}^{|\beta_k|}$, $k = 1, \ldots, l$. This problem can be solved using Spingarn's method. At each iteration we alternate between projection on the subspace defined by the consistency equations in (6.3) and evaluation of a prox-operator, by solving

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{k=1}^{l} \sum_{(i,j)\in W_k} |\,\mathbf{tr}(F_{ij}\mathcal{E}^*_{\beta_k}(\tilde{X}_k)) - \widehat{D}_{ij}| + (\sigma/2) \sum_{k=1}^{l} \|\tilde{X}_k - Z_k\|_F^2 \\ \text{subject to} & \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l. \end{array} \tag{6.4}$$

This problem is separable because if $(i,j) \in W_k$, F_{ij} is nonzero only in positions that are included in $\beta_k \times \beta_k$. The problems (6.4) can be solved efficiently via a straightforward modification of the interior-point method described in section 5.2.

We now illustrate the convergence of the decomposition method on two randomly generated networks. The nodes in the network are placed randomly using a uniform

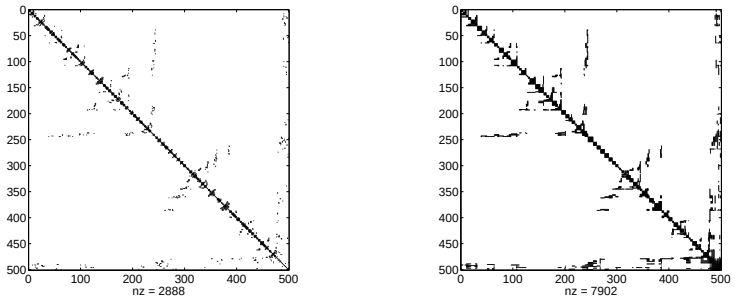

FIG. 6.1. *Sparsity pattern for a network with 500 nodes after AMD reordering and chordal embedding (left), and after clique merging (right). Before clique merging, there are 359 cliques with an average of 5 elements. After clique merging, there are 79 cliques with an average of 10 elements.*

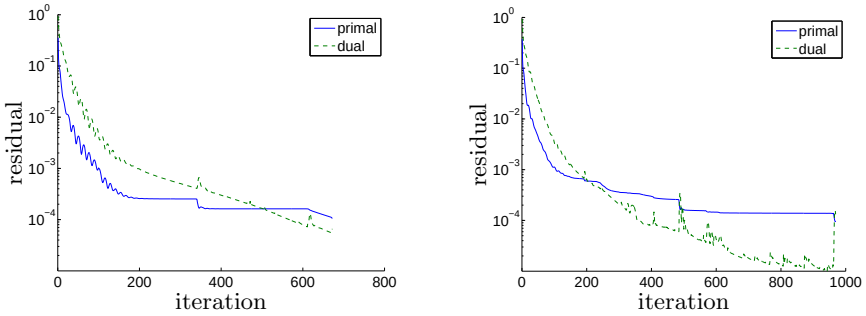

FIG. 6.2. *Relative primal and dual residuals versus iteration number for networks with 500 (left) and 2000 (right) nodes. For $n = 500$, there are 82 cliques, and for $n = 2000$, there are 310 cliques. A constant steplength parameter $\sigma_k = 5$ and relaxation parameter $\rho_k = 1$ are used.*

distribution on $[0, 1]^2$. The edges are assigned using the following rule: a pair (i, j) is in the sparsity pattern W if one of the nodes is among the five nearest neighbors of the other node. To compute a chordal embedding V, we use an approximate minimum degree (AMD) reordering (Figure 6.1, left). Often, the resulting embedding contains many small cliques and it is more efficient to merge some neighboring cliques, using algorithms similar to those in [6, 35, 23]. Specifically, traversing the tree in a topological order, we greedily merge clique k with its parent if $(|\beta_{\mathrm{pa}(k)}| - |\eta_k|)(|\beta_k| - |\eta_k|) \leq t_{\mathrm{fill}}$ or $\max(|\beta_k| - |\eta_k|, |\beta_{\mathrm{pa}(k)}| - |\eta_{\mathrm{pa}(k)}|) \leq t_{\mathrm{size}}$ where t_{fill} is a threshold based on the amount of fill that results from merging clique k with its parent, and t_{size} is a threshold based on the cardinality of the sets $\beta_{\mathrm{pa}(k)} \setminus \eta_{\mathrm{pa}(k)}$ and $\beta_k \setminus \eta_k$. In Figure 6.1 (right) we show the result of this technique using the values $t_{\mathrm{fill}} = t_{\mathrm{size}} = 5$.

A typical convergence plot of the resulting problem is given in Figure 6.2 for a network with 500 nodes (left) and 2000 nodes (right). A constant steplength parameter $\sigma_k = 5$ and relaxation parameter $\rho_k = 1$ are used. The greedy clique merging strategy described above was used, with the same threshold values.

6.2. Block-arrow semidefinite programs. In the second experiment we compare the efficiency of the splitting method with general-purpose SDP solvers. We consider a family of randomly generated SDPs with a block-arrow aggregate sparsity pattern V and a block-diagonal correlative sparsity pattern. The sparsity pattern V is defined in Figure 6.3. It consists of l diagonal blocks of size $d \times d$, plus w dense final

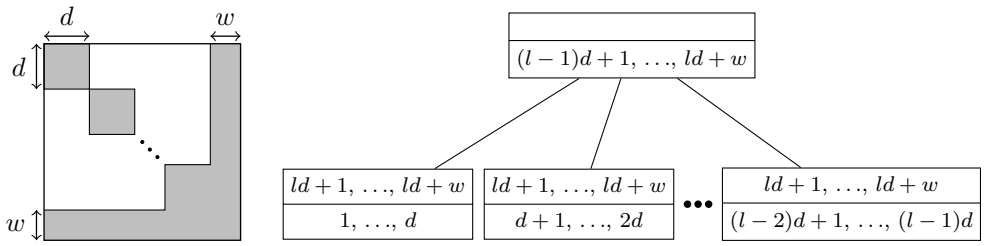

FIG. 6.3. *Block arrow pattern with l cliques and corresponding clique tree. The order of the matrix is $ld + w$. The first l diagonal blocks in the matrix have size d, the last block column and block row have width w. The cliques therefore have size $d + w$. Each clique in the clique tree is partitioned in two sets: the top row shows $\eta_k = \beta_k \cap \mathrm{pa}(\beta_k)$; the bottom row shows $\beta_k \setminus \eta_k$.*

rows and columns. We generate matrix cone LPs (4.2) with $m = ls$ primal equality constraints, partitioned in l sets $\nu_k = \{(k-1)s+1,\ (k-1)s+2,\ \ldots,\ ks\}$, $k = 1, \ldots, l$, of equal size $|\nu_k| = s$. If $i \in \nu_k$, then the coefficient matrix F_i contains a dense $\beta_k \times \beta_k$ block, and is otherwise zero. We will use the notation

$$(F_i)_{\beta_k\beta_k} = \begin{bmatrix} A_i & B_i \\ B_i^T & D_i \end{bmatrix},$$

for the nonzero block of F_i if $i \in \nu_k$. The primal and dual SDPs can be written as

$$\begin{array}{lll} \text{minimize} & \mathbf{tr}(CX) & \text{maximize} \quad b^T y \\ \text{subject to} & \mathcal{A}(X) = b & \text{subject to} \quad \mathcal{A}^*(y) + S = C \\ & X \succeq 0 & \qquad\qquad S \succeq 0 \end{array} \tag{6.5}$$

with a linear mapping $\mathcal{A} : \mathbf{S}^{(ld+w)\times(ld+w)} \to \mathbf{R}^{ls}$ defined as

$$\mathcal{A}(X)_i = \mathbf{tr}\left(\begin{bmatrix} A_i & B_i \\ B_i^T & D_i \end{bmatrix}\begin{bmatrix} X_{kk} & X_{k,l+1} \\ X_{l+1,k} & X_{l+1,l+1} \end{bmatrix}\right), \quad i \in \nu_k, \quad k = 1, \ldots, l,$$

where X_{ij} is the i, j block of X. (The dimensions are: $X_{l+1,l+1} \in \mathbf{S}^w$, $X_{l+1,i} \in \mathbf{R}^{w\times d}$, and $X_{ii} \in \mathbf{S}^d$ for $i = 1, \ldots, l$.) The adjoint $\mathcal{A}^*$ maps a vector $y \in \mathbf{R}^{ls}$ to the matrix

$$\mathcal{A}^*(y) = \begin{bmatrix} \sum_{i\in\nu_1} y_i A_i & 0 & \cdots & 0 & \sum_{i\in\nu_1} y_i B_i \\ 0 & \sum_{i\in\nu_2} y_i A_i & \cdots & 0 & \sum_{i\in\nu_2} y_i B_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \sum_{i\in\nu_l} y_i A_i & \sum_{i\in\nu_l} y_i B_i \\ \sum_{i\in\nu_1} y_i B_i^T & \sum_{i\in\nu_2} y_i B_i^T & \cdots & \sum_{i\in\nu_l} y_i B_i^T & \sum_{i=1}^m y_i D_i \end{bmatrix}.$$

In the reformulated problem, the variable X is replaced with l matrices $\tilde{X}_k = X_{\beta_k\beta_k}$, *i.e.*, defined as

$$\tilde{X}_k = \begin{bmatrix} (\tilde{X}_k)_{11} & (\tilde{X}_k)_{12} \\ (\tilde{X}_k)_{21} & (\tilde{X}_k)_{22} \end{bmatrix} = \begin{bmatrix} X_{kk} & X_{k,l+1} \\ X_{k,l+1}^T & X_{l+1,l+1} \end{bmatrix}, \quad k = 1, \ldots, l,$$

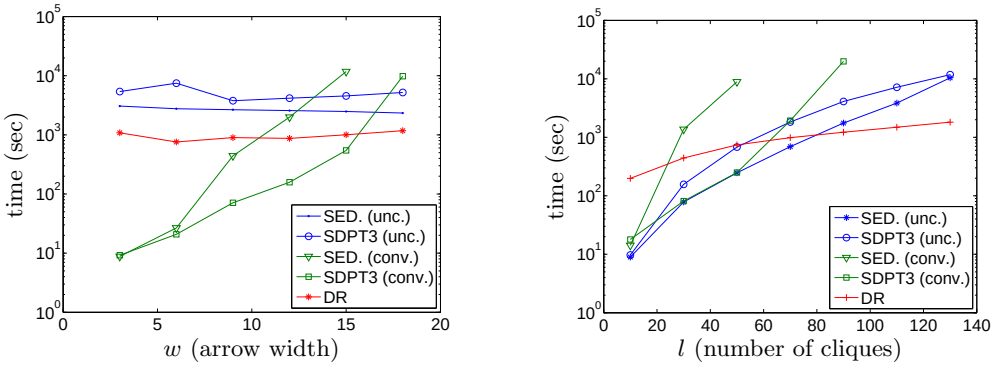

FIG. 6.4. *Solution time for randomly generated SDPs with block-arrow sparsity patterns. Times are reported for SEDUMI (SED.) and SDPT3 applied to the original ('unc.') and converted ('conv') SDPs, and the Spingarn method ('DR') applied to the converted SDP. The figure on the left shows the times as function of arrow width w, for fixed dimensions $l = 100$, $d = 20$, $s = 10$. The figure on the right shows the times versus number of cliques l, for fixed dimensions $w = 20$, $d = 20$, $s = 10$.*

and the primal SDP is converted to

$$
\begin{array}{lll}
& \text{min.} & \sum_{k=1}^{l} \mathbf{tr}(\tilde{C}_k \tilde{X}_k) \\
(6.6) & \text{s.t.} & \mathbf{tr}\left(\begin{bmatrix} A_i & B_i \\ B_i^T & D_i \end{bmatrix} \begin{bmatrix} (\tilde{X}_k)_{11} & (\tilde{X}_k)_{12} \\ (\tilde{X}_k)_{21} & (\tilde{X}_k)_{22} \end{bmatrix} \right) = b_i, \quad i \in \nu_k, \quad k = 1, \ldots, l \\
& & (\tilde{X}_k)_{22} = (\tilde{X}_l)_{22}, \quad k = 1, \ldots, l-1 \\
& & \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l
\end{array}
$$

where

$$
\tilde{C}_k = \begin{bmatrix} C_{kk} & C_{k,l+1} \\ C_{k,l+1}^T & 0 \end{bmatrix}, \quad k = 1, \ldots, l-1, \qquad \tilde{C}_l = \begin{bmatrix} C_{ll} & C_{l,l+1} \\ C_{l,l+1}^T & C_{l+1,l+1} \end{bmatrix}.
$$

With this choice of parameters, the correlative sparsity pattern of the converted SDP (6.6) is block-diagonal, *i.e.*, except for the consistency constraints $(\tilde{X}_k)_{22} = (\tilde{X}_l)_{22}$ the problem is separable with independent variables $\tilde{X}_k \in \mathbf{S}^{d+w}$. This allows us to compute the prox-operator by solving l independent conic QPs.

The problem data are randomly generated as follows. First, the entries of A_k, B_k, D_k are drawn independently from a normal distribution $\mathcal{N}(0,1)$. A strictly primal feasible X is constructed as $X = W + \alpha I$ where $W \in \mathbf{S}_V^{ld+w}$ is randomly generated with i.i.d. entries from $\mathcal{N}(0,1)$ and α is chosen so that $X_{\beta_k\beta_k} = W_{\beta_k\beta_k} + \alpha I \succ 0$ for $k = 1, \ldots, l$. The right-hand side b in the primal constraint is computed as $b_i = \mathbf{tr}(F_i X)$, $i = 1, \ldots, m$. Next, strictly dual feasible $y \in \mathbf{R}^m$, $S \in \mathbf{S}_V^{ld+w}$ are constructed. The vector y has i.i.d. entries from $\mathcal{N}(0,1)$ and S is constructed as $S = \sum_{k=1}^{l} \mathcal{E}_{\beta_k}^*(\tilde{S}_k)$, with $\tilde{S}_k = W_k + \alpha I$, $W_k \in \mathbf{S}^{|\beta_k|}$ randomly generated with i.i.d. $\mathcal{N}(0,1)$ entries, and α chosen so that $\tilde{S}_k \succ 0$. The matrix C is constructed as $C = S + \sum_i y_i F_i$.

In Figure 6.4 we compare the solution time of Spingarn's algorithm with the general-purpose interior-point solvers SEDUMI and SDPT3, applied to the unconverted and converted SDPs (4.1) and (5.1). In the decomposition method we use a constant steplength parameter $\sigma_k = 5$ and relaxation parameter $\rho_k = 1.75$. The stopping criterion is (3.20) with $\epsilon_{\mathrm{p}} = \epsilon_{\mathrm{d}} = 10^{-4}$. For each data point we report the average

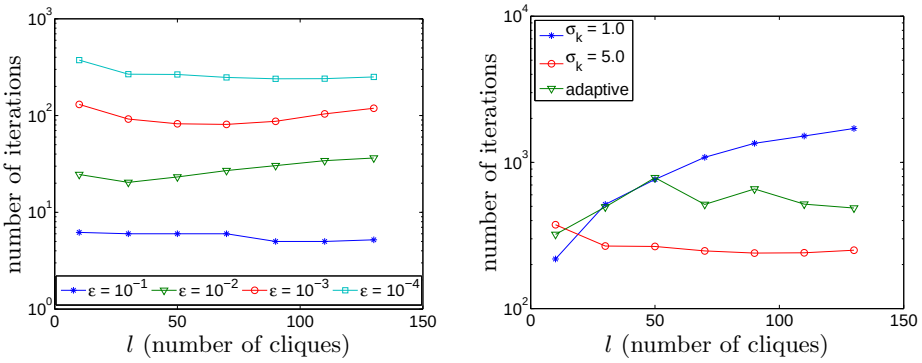

FIG. 6.5. Left. *Number of iterations for of Spingarn's method based on the desired accuracy, for a problem instance with* $d = 20$, $w = 20$, $s = 10$, *and using a fixed steplength parameter* $\sigma_k = 5$. Right. *Number of iterations for the same problem with* $\epsilon_{\mathrm{p}} = \epsilon_{\mathrm{d}} = 10^{-4}$ *and different choices of steplength.*

CPU time over 5 instances. To interpret the results, it is useful to consider the linear algebra complexity per iteration of each method. The unconverted SDP (6.5) has a single matrix variable X of order $p = ld + w$. The cost per iteration of an interior-point method is dominated by the cost of forming and solving the Schur complement equation, which is dense and of size $m = sl$. For the problem sizes used in the figures (w small compared to ld) the cost of solving the Schur complement dominates the overall complexity. This explains the nearly constant solution time in the first figure (fixed l, s, p, varying w) and the increase with l shown in the second figure. The converted SDP (6.6) on the other hand has l variables $\tilde{X}_k$ of order $d + w$. The Schur complement equation in an interior-point method has the general structure (3.11) with a leading block-diagonal matrix (l blocks of size $s \times s$) augmented with a dense block row and block column of width proportional to lw^2. For small w, exploiting the block-diagonal structure in the Schur complement equation allows one to solve the Schur complement equation very quickly and reduces the cost per iteration to a fraction of a cost of solving the unconverted problem, despite the increased size of the problem. However the advantage disappears with increasing w (Figure 6.4 left). The main step in each iteration of the Spingarn method applied to the converted problem is the evaluation of the prox-operators via an interior-point method. The Schur complement equations that arise in this computation are block-diagonal (l blocks of order s) and therefore the cost of solving them is independent of w and linear in l. As an additional advantage, since the correlative sparsity pattern is block-diagonal, the proximal operator can be evaluated by solving l independent conic QPs that can be solved in parallel. This was not implemented in the experiment, but could reduce the solution time by a factor of roughly l.

The principal disadvantage of the splitting method, compared with an interior-point method, is the more limited accuracy and the higher sensitivity to the choice of algorithm parameters. Figure 6.5 (left) shows the number of iterations versus l for different values of the tolerance ϵ in the stopping criterion. The right-hand plot shows the number of iterations versus l for two different constant values of the steplength parameter σ_k ($\sigma_k = 1.0$ and $\sigma_k = 5.0$) and for an adaptively adjusted steplength.

So far we have assumed that the error in the proximal operator evaluations is negligible compared with the exit tolerances used in Spingarn's method. In the last

ϵ_{prox}	$\epsilon_{\text{p}} = \epsilon_{\text{d}}$	r_X	r_y	r_{obj}	#iterations
10^{-6}	10^{-4}	$9.4\ 10^{-3}$	$7.6\ 10^{-4}$	$1.6\ 10^{-5}$	179
10^{-8}	10^{-4}	$9.1\ 10^{-3}$	$7.2\ 10^{-4}$	$1.5\ 10^{-5}$	180
	10^{-6}	$1.2\ 10^{-4}$	$8.6\ 10^{-6}$	$5.7\ 10^{-8}$	443
	10^{-8}	$2.1\ 10^{-5}$	$2.1\ 10^{-6}$	$1.8\ 10^{-8}$	2376
10^{-10}	10^{-4}	$9.1\ 10^{-3}$	$7.2\ 10^{-4}$	$1.5\ 10^{-5}$	180
	10^{-6}	$1.2\ 10^{-4}$	$8.2\ 10^{-6}$	$4.3\ 10^{-8}$	444
	10^{-8}	$1.1\ 10^{-5}$	$2.8\ 10^{-7}$	$1.9\ 10^{-8}$	759

TABLE 6.1

Relative differences between solutions, computed by Spingarn's method and an interior-point method, and the number of iterations in Spingarn's method, for varying exit conditions in Spingarn's method and the prox-operator evaluations. The first column is the tolerance in the prox-operator evaluations. The second column shows the tolerances in Spingarn's method.

experiment we examine the effect of inexactness of the prox-operator evaluations. As test problem we use an instance of the family of block-arrow problems, with $d = 10$, $l = 25$, $w = 10$, $s = 10$. The accuracy of the conic QP solver used to evaluate the prox-operators is controlled by three tolerances that bound the error in the optimality conditions (3.24). The tolerance ϵ_{feas} is an upper bound on the relative error in the primal and dual equality constraints, ϵ_{abs} is an upper bound on the duality gap, and ϵ_{rel} is an upper bound on the relative duality gap. In the experiment we set these tolerances to $\epsilon_{\text{feas}} = \epsilon_{\text{abs}} = \epsilon_{\text{rel}}/10 = \epsilon_{\text{prox}}/10$, for three values of ϵ_{prox}: the CVXOPT default value $\epsilon_{\text{prox}} = 10^{-6}$, and two smaller values, 10^{-8} and 10^{-10}. For each value, we run Spingarn's method with different tolerances ϵ_{p} and ϵ_{d} in the stopping condition (3.20). The results are shown in Table 6.1. In columns 3–5 we compare the solution from Spingarn's method with the answer returned by an interior-point solver (SDPT3). The entries in these columns are defined as

$$r_X = \frac{\|\tilde{X} - \tilde{X}_{\text{ipm}}\|_F}{\|\tilde{X}_{\text{ipm}}\|_F}, \qquad r_y = \frac{\|y - y_{\text{ipm}}\|_2}{\|y_{\text{ipm}}\|_2}, \qquad r_{\text{obj}} = \frac{|f(\tilde{X}) - f(\tilde{X}_{\text{ipm}})|}{|f(\tilde{X}_{\text{ipm}})|}$$

where $\tilde{X}_{\text{ipm}}$ and y_{ipm} are the optimal primal and dual variables of the converted problem computed by SDPT3, and the function $f(\tilde{X})$ is the primal objective value. The last column in the table is the number of iterations in Spingarn's method. (For $\epsilon_{\text{prox}} = 10^{-6}$ and $\epsilon_{\text{p}} = \epsilon_{\text{d}} = 10^{-6}$, the method did not converge in 10000 iterations.)

From the table we can make a few observations. First, when $\epsilon_{\text{prox}} \ll \epsilon_{\text{p}} = \epsilon_{\text{d}}$, the relative error in the solution seems to be comparable in magnitude with the primal and dual residuals. Second, when running Spingarn's method with a moderate accuracy (for example, with $\epsilon_{\text{p}} = \epsilon_{\text{d}} = 10^{-4}$), increasing the accuracy of the prox-operator evaluation does not improve the convergence or the accuracy of the result, and the accuracy of an interior-point method with typical default settings is adequate.

7. Conclusions. We have described a decomposition method that exploits partially separable structure in linear conic optimization problems. The basic idea is straightforward: by replicating some of the variables, we reformulate the problem as an equivalent linear optimization problem with block-separable conic inequalities and an equality constraint that ensures that the replicated variables are consistent. We can then apply Spingarn's method of partial inverses to this equality-constrained convex problem. Spingarn's method is a generalized alternating projection method for convex optimization over a subspace. It alternates orthogonal projections on the subspace with the evaluation of the proximal operator of the cost function. In the

method described in the paper, these prox-operators are evaluated by an interior-point method for conic quadratic optimization.

When applied to sparse semidefinite programs, the reformulation coincides with the clique conversion methods which were introduced in [25, 16] with the purpose of exploiting sparsity in interior-point methods for semidefinite programming. By solving the converted problems via a splitting algorithm instead of an interior-point algorithm we extend the applicability of the conversion methods to problems for which the converted problem is too large to handle by interior-point methods. As a second advantage, if the correlative sparsity is block-diagonal, the most expensive step of the decomposition algorithm (evaluating the proximal operator) is separable and can be parallelized. The numerical experiments indicate that the approach is effective when a moderate accuracy (compared with interior-point methods) is acceptable. However the convergence can be quite slow and strongly depends on the choice of steplength.

A critical component in the decomposition algorithm for semidefinite programming is the use of a customized interior-point method for evaluating the proximal operators. This technique allows us to evaluate the proximal operator at roughly the same cost of solving the reformulated SDP without the consistency constraints. As a further improvement we hope to extend this technique to exploit sparsity in the coefficient matrices of the reformulated problem, using techniques developed for interior-point methods for sparse matrix cones [5].

REFERENCES

[1] J. Agler, J. W. Helton, S. McCullough, and L. Rodman, *Positive semidefinite matrices with a given sparsity pattern*, Linear Algebra and Its Applications, 107 (1988), pp. 101–149.

[2] M. Andersen, J. Dahl, and L. Vandenberghe, *CVXOPT: A Python Package for Convex Optimization*, `www.cvxopt.org`, 2010.

[3] M. S. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe, *Interior-point methods for large-scale cone programming*, in Optimization for Machine Learning, S. Sra, S. Nowozin, and S. J. Wright, eds., MIT Press, 2012, pp. 55–83.

[4] M. S. Andersen, J. Dahl, and L. Vandenberghe, *Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones*, Mathematical Programming Computation, 2 (2010), pp. 167–201.

[5] ———, *Logarithmic barriers for sparse matrix cones*, Optimization Methods and Software, 28 (2013), pp. 396–423.

[6] C. Ashcraft and R. Grimes, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Transactions on Mathematical Software, 15 (1989), pp. 291–309.

[7] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, 2011.

[8] J. R. S. Blair and B. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, 1993.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of mulitipliers*, Foundations and Trends in Machine Learning, 3 (2011), pp. 1–122.

[10] S. Burer, *Semidefinite programming in the space of partial positive semidefinite matrices*, SIAM Journal on Optimization, 14 (2003), pp. 139–172.

[11] A. M.-C. Cho and Y. Ye, *Theory of semidefinite programming for sensor network localization*, Mathematical Programming, Series B, 109 (2007), pp. 367–384.

[12] P. L. Combettes and J.-C. Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE Journal of Selected Topics in Signal Processing, 1 (2007), pp. 564–574.

[13] E. Dall'Anese, H. Zhu, and G. B. Giannakis, *Distributed optimal power flow for smart microgrids*, (2012). `arxiv.org/1211.5856`.

[14] J. Eckstein, *Parallel alternating direction multiplier decomposition of convex programs*, Journal of Optimization Theory and Applications, 80 (1994), pp. 39–62.

[15] J. Eckstein and D. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55 (1992), pp. 293–318.
[16] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, *Exploiting sparsity in semidefinite programming via matrix completion I: general framework*, SIAM Journal on Optimization, 11 (2000), pp. 647–674.
[17] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, eds., Springer, 2008, pp. 95–110.
[18] ———, *CVX: Matlab Software for Disciplined Convex Programming, version 2.0 (beta)*, `cvxr.com`, 2012.
[19] A. Griewank and P. L. Toint, *Partitioned variable metric updates for large structured optimization problems*, Numerische Mathematik, 39 (1982), pp. 119–137.
[20] ———, *On the existence of convex decompositions of partially separable functions*, Mathematical Programming, 28 (1984), pp. 25–49.
[21] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz, *Positive definite completions of partial Hermitian matrices*, Linear Algebra and its Applications, 58 (1984), pp. 109–124.
[22] B. S. He, L. Z. Liao, and S. L. Wang, *Self-adaptive operator splitting methods for monotone variational inequalities*, Numerische Mathematik, 94 (2003), pp. 715–737.
[23] J. Hogg and J. Scott, *A modern analyse phase for sparse tree-based direct methods*, Tech. Rep. RAL-TR-2010-031, Rutherford Appleton Laboratory, Didcot, UK, 2010.
[24] N. Kakimura, *A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices*, Linear Algebra and Its Applications, 433 (2010), pp. 819–823.
[25] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita, *Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion*, Mathematical Programming, 129 (2011), pp. 33–68.
[26] S. Kim, M. Kojima, and H. Waki, *Exploiting sparsity in SDP relaxations for sensor network localization*, SIAM Journal on Optimization, 20 (2009), pp. 192–215.
[27] K. Kobayashi, S. Kim, and M. Kojima, *Correlative sparsity in primal-dual interior-point methods for LP, SDP, and SOCP*, Applied Mathematics and Optimization, 58 (2008), pp. 69–88.
[28] L. S. Lasdon, *Optimization Theory for Large Systems*, Dover Publications, Inc., 2002. First published in 1970 by the MacMillan Company.
[29] J. G. Lewis, B. W. Peyton, and A. Pothen, *A fast algorithm for reordering sparse matrices for parallel factorization*, SIAM Journal on Scientific and Statistical Computing, 10 (1989), pp. 1146–1173.
[30] Z. Lu, A. Nemirovski, and R. D. C. Monteiro, *Large-scale semidefinite programming via a saddle-point Mirror-Prox algorithm*, 109 (2007), pp. 211–237.
[31] J. J. Moreau, *Proximité et dualité dans un espace hilbertien*, Bull. Math. Soc. France, 93 (1965), pp. 273–299.
[32] K. Nakata, K. Fujitsawa, M. Fukuda, M. Kojima, and K. Murota, *Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical details*, Mathematical Programming Series B, 95 (2003), pp. 303–327.
[33] N. Parikh and S. Boyd, *Graph projection block splitting for distributed optimization*, Mathematical Programming Computation, 6 (2014), pp. 77–102.
[34] A. Pothen and C. Sun, *Compact clique tree data structures in sparse matrix factorizations*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., Society for Industrial and Applied Mathematics, 1990, pp. 180–204.
[35] J. K. Reid and J. A. Scott, *An out-of-core sparse cholesky solver*, ACM Transactions on Mathematical Software, 36 (2009), p. 133.
[36] R. T. Rockafellar, *Convex Analysis*, Princeton Univ. Press, Princeton, second ed., 1970.
[37] J. E. Spingarn, *Partial inverse of a monotone operator*, Applied Mathematics and Optimization, 10 (1983), pp. 247–265.
[38] ———, *Applications of the method of partial inverses to convex programming: decomposition*, Mathematical Programming, 32 (1985), pp. 199–223.
[39] G. Srijuntongsiri and S. Vavasis, *A fully sparse implementation of a primal-dual interior-point potential reduction method for semidefinite programming*, (2004). `arXiv:cs/0412009`.
[40] J. F. Sturm, *Using SEDUMI 1.02, a Matlab toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11-12 (1999), pp. 625–653.
[41] K. C. Toh, R. H. Tütüncü, and M. J. Todd, *Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems*, Pacific Journal of Optimization, 3 (2007).